# THE DEVELOPMENT OF AN EXPERT SYSTEM FOR SOFTWARE COST ESTIMATION

THESIS

James L. Goodson, Major, USAF

AFIT/GLM/LSM/90S-21



DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

2

AFIT/GLM/LSM/90S-21



THE DEVELOPMENT OF AN EXPERT SYSTEM
FOR SOFTWARE COST ESTIMATION

THESIS

James L. Goodson, Major, USAF

AFIT/GLM/LSM/90S-21

The opinions and conclusions in this paper are those of the author and are not intended to represent the official position of the DOD, USAF, or any other government agency.

AFIT/GLM/LSM/90S-21

# THE DEVELOPMENT OF AN EXPERT SYSTEM FOR SOFTWARE COST ESTIMATION

THESIS

Presented to the Faculty of the School of Systems and Logistics
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Logistics Management

James L. Goodson, B.S., M.S.S.M.
Major, USAF

September 1990

## Trademark Acknowledgements

Nexpert and Nexpert Object are trademarks of Neuron Data, Incorporated.

PRICE-S is a trademark of General Electric, Incorporated.

SEER is a trademark of Galorath Associates, Incorporated.

## Table of Contents

Page

## List of Figures

## List of Tables

## List of Equations

AFIT/GLM/LSM/90S-21

## Abstract

Software development has become an integral part of new weapon system design as sophisticated computer operating and control systems have proliferated. Concurrently, there has been an ever increasing requirement to accurately forecast system development cost.

Experts use complicated software cost estimating tools which are difficult for novice analysts to use. Expert system technology appears to be able to bridge this gap.

Policy makers have become intrigued by the potential offered by the development of expert systems in today's weapon systems for system support and development and operational utility.

The primary objective of this thesis was to gather expert knowledge in software cost estimation and integrate it with a powerful analytical software cost estimation algorithm. A battery of questions where given to the experts to elicit responses relative to their knowledge in software cost estimation. Responses where integrated with algorithms with the detailed COCOMO cost estimation model.

The expert system designed in this ~~research~~ thesis provides software developers, program managers, and cost analysts an easy mechanism for determining software development costs. It provides an intelligent preprocessor, numeric algorithms and an intelligent post processor in one tool. The expert system can help the novice make accurate estimates and speed the process for experts.

THE DEVELOPMENT OF AN EXPERT SYSTEM FOR SOFTWARE COST ESTIMATION

## I. Introduction

### Cost of Software Development

The Air Force faces a large reduction in defense spending. Concurrent with that reduction is a continued requirement for the development of new weapon systems to replace aging and hard to maintain older systems. A major portion of the cost for these acquisitions, estimated to be $25 billion for fiscal year 1990, is computer software development (Ferens, 1990:vii).

As technology advances, policy makers have become intrigued by the potential offered by the development of Artificial Intelligence (AI) in today's weapon systems both for its operational utility and for system support and development (Valusek, 1990). The application of Expert System technology, a subset of AI, is of particular interest in its application in the estimation of software development costs.

### Manpower

Budget reductions dictate that sacrifices be made in all areas of management including manpower. With the reduction in manpower and the potential exodus of qualified personnel, "experts", out of the service, new methods must be developed to ensure that novices will be able to make expert decisions on problems in which they have little or no expertise. In his preface to the student in his book Software Engineering Economics, Barry W. Boehm cites the following concern:

> There is a good chance that, within a few years, you will find yourself together in a room with a group of people who will be deciding how much time and money you should get to do a significant new software job. Perhaps one or two of the people in the room will know software well, but most of them will not.
>
> (Boehm, 1981:xxxiii)

Specific Problem

With the present level of expertise diminishing in the Air Force, it is now incumbent on people with limited training in cost estimation,like engineers and logisticians, to do software cost analysis. Present software estimation models rely on extensive training and/or experience to use them effectively. The models all require the user to have a clear understanding of the lexicon of software cost estimation in addition to those terms that are specific to the model in use (McMurry and Nelson, 1990:2-11). A method is needed to capture expert software cost estimation knowledge before it is lost.

Possible Solution

Expert system technology has been applied in many fields including inventory management (Allen, 1989), medical diagnosis, and the troubleshooting of malfunctions in mechanical and electronic systems (Hicks, 1988:5-10). Additionally, expert systems have been created to provide problem solving guidance in managing budgets, cost estimation, monitoring local area networks, and advising emergency response teams (Nexpert Partners, 1989:2). Given that expert systems have been built using similar problem solving logic as software cost estimation, it seems realistic to assume that a rule-based system using cognitive decision making criteria can be of significant benefit to the analysis and the definition of software cost estimation. This thesis will analyze how software cost estimation is presently being done, how to acquire software cost estimation expert knowledge, develop an expert system to aid program managers in the acquisition of new software and upgrades to existing software and compare the expert system to current methods. Better estimates could reduce software development costs, manpower, and decrease the time it takes to place a new effort on contract.

## Investigative Questions

The following questions will be investigated to support the thesis objectives stated in the above paragraph:

1. What knowledge and heuristics, rules of thumb, do software cost analysts use in decision making?

2. Can this knowledge and these heuristics can be captured and programmed as an "expert system."

3. Do expert system significantly improve the performance of software cost analysts.

4. Can the use of expert systems allow cost analysts to solve complex or difficult problems more efficiently?

## Scope

The following factors define the scope of this research effort:

1. The cost models selected for comparison to the expert system under development were chosen for their general acceptance Air Force wide and their present use in the Advanced Tactical Fighter System Program Office.

2. The selection of the expert system shell was predicated on its availability, its compatibility with a Macintosh computer system, and its transportability to other C based, object oriented computer systems.

3. Each cost model and the expert system will be exercised using the same case study to ensure a product baseline for comparison.

4. Validation of the expert system will be done on case files provided by the ATF SPO if they can be released to the researcher.

5. At least one expert from a system program office and one from outside the Aeronautical Systems Division will be interviewed to preclude the potential of institutional ideas being integrated into the expert system data base.

Limiting Factor

Only non-proprietary, public domain, algorithms will be used in the development of this cost estimation expert system.

Thesis Structure

Chapter one identifies the purpose and direction of this effort. Chapter II covers the background, history, a review of parametric cost estimating models, and a review of the expert shell that is used for the development of this system. Chapter III identifies the method for solving the problem and the method for evaluation and validation of the proposed expert system. Chapter IV contains research results. Chapter V makes further conclusions and recommendations.

# II. Literature Review

## Overview

This chapter contains a review of the literature and the background and history of expert system development. It also covers software cost economics, the software cost estimating models used in this research, and a review of expert system software cost estimation.

## Artificial Intelligence and Expert Systems

Artificial intelligence (AI) is defined as, "The study of how to make computers do things that previously only people could do (Valusek, 1990)." Or in other words, AI programs into the computer the cognitive decision making processes that are an inherent part of the human manipulation of knowledge. AI is represented by the following terms: heuristics or rules of thumb; natural or human language processing; advanced robotics; object oriented programming; symbolic computation; and expert systems (ES) to name a few (Valusek, 1990 and Ferens, 1990a:19-5).

Artificial intelligence, and expert systems in particular, are difficult entities to understand and use. The following figure will help place perspective on how AI compares with the algorithmic approach of programming and problem solving. Note that in AI systems there is a recognition that an optimal or best solution may not be possible and that a satisfactory solution is all that is required.

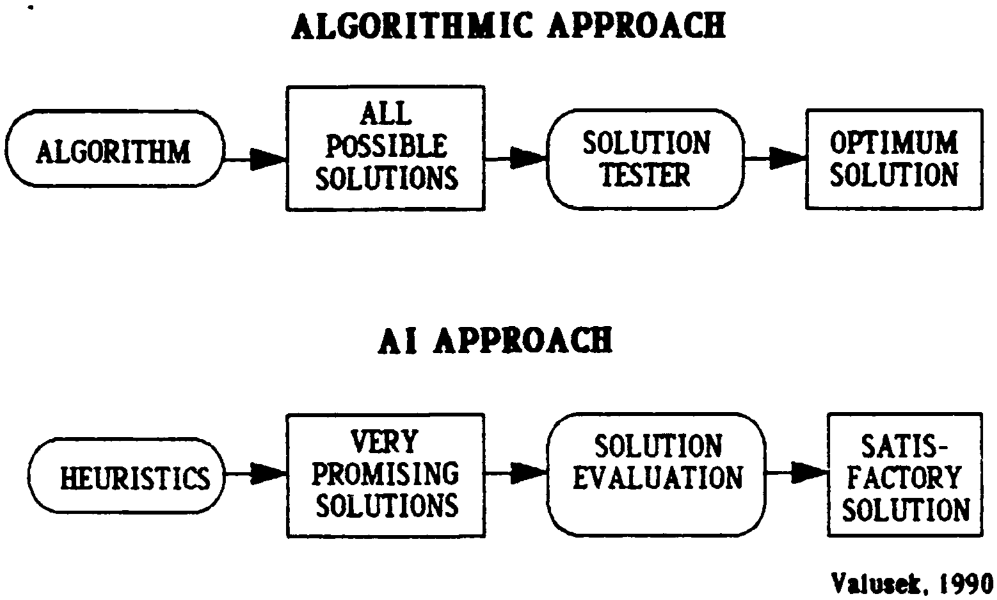


Figure 1. Algorithmic Vs. AI Problem Solving.

An expert system is a computer program that exploits expert knowledge, solves difficult problems, and uses understandable reasoning procedures. It must provide to the user a high degree of expertise, predictive modeling power, a storehouse of expert knowledge, and a tutoring facility (Valusek, 1990). Douglas Waterman goes on to say that an ES must have:

| | |
|---|---|
| Expertise | - Exhibit expert performance<br>- A high level of skill<br>- Have adequate robustness |
| Symbolic Reasoning | - Represent knowledge symbolically<br>- Reformulate symbolic knowledge |
| Depth | - Handle difficult problem domains<br>- Use complex rules |
| Self-knowledge | - Examine its own reasoning<br>- Explain its operation |

(Waterman, 1986:25)

Expert system are made of six basic components with interfaces for system and knowledge design, and for the integration of end user requirements. The six components as depicted in Figure 6 are: the knowledge base where rules and facts reside, the inference engine which defines the search and control mechanisms, a knowledge acquisition subsystem that provides the tools for knowledge integration, an explanation subsystem that provides the how and why of particular operations and functions, a working memory, and a user interface.

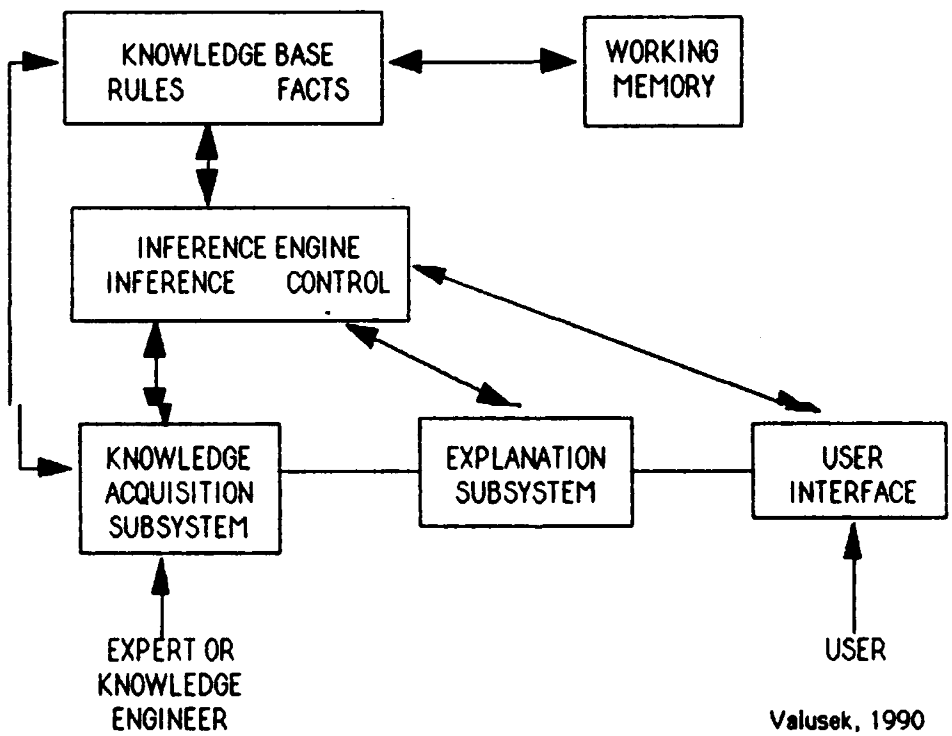


Figure 2. Components of An Expert System

These components form an integrated unit each interacting with the other.

Before building an expert system one must evaluate whether the benefits to be gained from the effort outweigh the problems that are an intrinsic part of developing any new technology. Holt and Valusek identify a variety of benefits and cautions that are an intrinsic part of using expert systems for problem solving and decision aiding. They compare human expertise to artificial expertise and identify the good and bad of each.

Table 1

The Good and the Bad News for Artificial Expertise
(Holt, 1990 and Valusek, 1990)

| The Good News for Artificial Expertise | |
|---|---|
| Human Expertise | Artificial Expertise |
| - Perishable | - Permanent |
| - Difficult to transfer | - Easy to transfer |
| - Difficult to document | - Easy to document |
| - Unpredictable | - Consistent |
| - Expensive | - Affordable |
| The Bad News for Artificial Expertise | |
| - Creative | - Uninspired (Systematic) |
| - Adaptive | - Needs to be told |
| - Sensory experience | - Symbolic input |
| - Broad focus | - Narrow focus |
| - Common sense knowledge | - Technical knowledge |

Expert systems are complex in their design and implementation. Managers must weigh the complexities of ES design and the benefits that can be derived from their use. Problems selected for expert system application should warrant the time, effort, and expense required for its development. David S. Prerau defined a set of criteria (Prerau, 1985:26-30) to test the appropriateness for expert system implementation. This lengthy criteria is in appendix B and is reviewed to see how software cost estimation fares as a candidate for expert system development. Table 3 summarizes the results.

## Table 2

### Summary of Prerau's Criteria Relative to Software Cost Estimation (See Appendix B)

BASIC REQUIREMENTS

The domain is characterized by the use of expert knowledge, judgment, and experience.

Conventional programming approaches to the task are not satisfactory.

There are recognized Experts that solve the problem today.

There is a need to capture the expertise.

TYPE OF PROBLEM

The task requires the use of heuristics, e.g., rules of thumb, strategies, etc.

The task is defined very clearly.

THE EXPERT

There exists an expert to work with the project.

The expert has built up expertise over a long period.

The expert is capable of communicating his knowledgeand experience.

The expert is cooperative and easy to work with.

PROBLEM BOUNDS

The problem is neither too easy nor too difficult.

The task is sufficiently narrow and self-contained.

The number of important concepts required is bound to several hundreds.

DOMAIN AREA PERSONNEL

Personnel in the domain area are realistic.

Domain area personnel understand the system will be limited in scope.

Potential users would welcome the completed system.

OTHER DESIRABLE FEATURES

The task is decomposable, allowing relatively rapid prototyping.

The skill required by the task is taught to novices.

Test cases are available.

The domain is fairly stable.

The user interface will not require extensive effort

Douglas Waterman identified five areas for which a candidate system or knowledge area should be evaluated when considering the development of an expert system. He tests to see if the knowledge area in question, is a 'realistic' candidate (that the knowledge is volatile and the problem persistent), is 'justified' by its payoff in savings of resources and other environmental factors, and has experts or 'expertise' that can be emulated by the expert system. He continues by testing to see if it has 'tasks' that are manageable, uses heuristics that are well understood, and satisfies 'other' criteria relative to cognitive operations inherent in expert decision making processes (Waterman, 1986:127-134). His assertion that an expert sytstem must meet these five criteria parallels the recommendations made by Prerau as listed in appendix B.

## Software Engineering

The software cost tools used in this study are described as top down parametric models. Top down implys that the software costs are calculated based on "design characteristics or parameters of a software program." The cost is computed for the entire system or computer software configuration item (CSCI) and then partitioned down to lower levels or development phases (Ferens, 1990a:10-3). In all cases the algorithms and equations used in the models are developed through regression analysis. The data bases are derived by correlating the historical data of several systems to performance characteristics of those same systems (Project, 1988:2).

## Cost Estimation Models

The following models form the basis of comparison for evaluating the expert system under development in this study. They were selected due to their wide acceptance among users in industry and the Air Force and that they have been validated as effective in software cost estimation (Ferens, 1990a).

### Constructive Cost Model (COCOMO)

To apply expert systems to software cost analysis it is important to have an understanding of how this analysis is done. Barry W. Boehm discusses this topic extensively in his book Software Engineering Economics. He describes the history of software cost economics, its impact on effective program management, and a cost estimation tool called COCOMO, or COnstructive COst MOdel (BOEHM, 1981:xx). COCOMO is a series of three models, basic, intermediate, and detailed whose use is dependent upon the complexity or level of detail required to accomplish an effective evaluation. The basic COCOMO is used when the analyst needs a rough order of magnitude estimate but has few factors available to make his estimate. The intermediate COCOMO includes factors in terms of their 'aggregate impact' on overall project cost. The rules and algorithms in the detailed COCOMO model describes the relationship between factors used in making the cost estimate and the influence those factors have on the project's development phases (Boehm,1981:58).

The first factor for the software cost estimator to consider is the development mode of his project. The modes, organic, semi-detached, and embedded, are predicated by the level of integration and the complexity the system is being designed. Software projects in a stable environment with a minimal need for creative architectures or algorithms, a low premium on early completion, and is of small size (50 thousand lines delivered source instructions (KDSI) or less) are called organic. A semi-detached mode has some or all of the organic mode in addition to larger size (up to 300 KDSI) and more rigid compliance with requirements and specifications. The embedded mode is the most rigorous by far in that it is quite large, user requirements are potentially volatile, and the effects of software failure could be catastrophic to equipment or human life (Boehm, 1981:78-80). The equations listed below are used to compute the man-months of effort and the schedule time it takes for that effort. The analyst reviews the software proposal and finds the estimated delivered source instructions for each software module or subystem. If, for example, he finds that a module is estimated to contain 10,000 lines of code and that the development effort is embedded, he can compute the man-months of effort and the development time by using equations 5 and 6. In this case, $(MM)_{NOM} = 2.8(10)^{1.20} = 44.37$ man-months of effort. Development time can be approximated by inserting the man-months of effort from equation 5 into equation 6: TDEV = $2.5(44.37)^{0.32} = 8.4$ calender months. See appendix E for other equations that are used in addition to these (Boehm, 1981:353).

| | | |
|---|---|---|
| Organic | $(MM)_{NOM} = 3.2(KDSI)^{1.05}$ | (1) |
| | $TDEV = 2.5(MM)_{DEV}^{0.38}$ | (2) |
| Semi-Detached | $(MM)_{NOM} = 3.0(KDSI)^{1.12}$ | (3) |
| | $TDEV = 2.5(MM)_{DEV}^{0.35}$ | (4) |
| Embedded | $(MM)_{NOM} = 2.8(KDSI)^{1.20}$ | (5) |
| | $TDEV = 2.5(MM)_{DEV}^{0.32}$ | (6) |

where,
$(MM)_{NOM}$ is man-months nominal,
KDSI is thousand of delivered source instructions,
TDEV is development schedule,
$(MM)_{DEV}$ is development time in man-months.

Most of the Air Force's software development projects are described as embedded due to their complexity, integration requirements, and rigid controls for their development.

In addition to development mode, Boehm defines four factors, product, computer, personnel, project, with fifteen subordinate attributes that the cost estimator must analyze and quantify. Unlike the basic and intermediate COCOMO models discussed earlier, the embedded mode is used to evaluate the factors in much finer detail by breaking the effort into modules and subsystems (see Table 3).

COCOMO rates each attribute on a continuum from very low to very high which changes as a function of the level of project development (see appendix E). The levels of development are: requirements analysis and preliminary design (RPD), detailed design (DD), code and unit test (CUT), and component integration and test (IT). Notice that Boehm does not consider a formal test for the entire effort which would occur after IT.

Table 3

Detailed COCOMO Factors by Module and Subsystem
(Boehm, 1981: 371-474)

| | |
|---|---|
| **Module Level Effort Multipliers** | |
| CPLX | Software product complexity |
| PCAP* | Programmer capability |
| VEXP* | Virtual machine experience |
| LEXP* | Programming language experience |
| **Subsystem Level Effort Multipliers** | |
| *Product* | |
| RELY | Required software reliability |
| DATA | Data Base Size |
| *Computer* | |
| TIME | Execution time constraint |
| STOR | Main storage constraint |
| VIRT | Virtual machine volatility |
| TURN | Computer turnaround time |
| *Personnel* | |
| ACAP* | Analyst capability |
| AEXP* | Applications experience |
| *Project* | |
| MODP | Use of modern programming practices |
| TOOL | Use of software tools |
| SCED | Development schedule constraint |

*Attributes that fall under Personnel factors.

Boehm also recommended a set of criteria that are advantageous to use in evaluating the different models for their utility in estimating software development cost (Boehm, 1981: 476). These criteria provide a qualitative continuum for comparing the different models and give the manager a mechanism for selecting the best or optimal model to use considering his application. The criteria are discussed in chapter three.

GE PRICE-S™. PRICE-S™ is a commercially available parametric cost model that was originally developed by Dr. Robert Park of RCA PRICE systems. It is a proprietary model in that all of the algorithms, data, and instruction sets are the property of General Electric. As a consequence, the software cannot be bought per se, but can be leased. The limitations of this proprietary relationship for the user are cost for the lease, training, and the inability to modify the instruction set to fit unique requirements (Ferens, 1990c).

PRICE-S™ provides life cycle cost analysis of the system design, programming, data, system engineering and program management, quality assurance, and configuration management. It provides these costs relative to the program development phase from system concept, preliminary design, detailed design, system test and evaluation, to operational test and evaluation. It also provides a system level development schedule with adjustments made for program acceleration, stretch-out, and phase transition inefficiencies (PRICE, 1989: 1-1). A more extensive discussion on input and output will be provided in chapters three and four of this paper.

Unlike COCOMO which has fifteen factors plus source lines of code for input variables, PRICE-S™ groups its inputs into seven fundamental categories (see table 4). The model is designed to be easily calibrated to fit programmatic changes plus tests the credibility and consistency of input data which helps to minimize input of faulty information (PRICE, 1989:1-8)."

Table 4

PRICE-S™ Input Variables
(PRICE-S, 1989:1-8)

1. Project Magnitude (How big?)
   The amount of code to be produced.
2. Program Application (What Character?)
   The type of project such as MIS, Communications, Telemetry, etc.
3. Level of New Design and Code (How much new work is needed?)
   The amount of new design and code that cannot be taken from the existing inventory.
4. Productivity (Who will do the work?)
   The experience, skill, and know-how of the assigned individuals or team, as applicable to the specified task.
5. Utilization (What hardware constraints?)
   The extent of processor loading relative to its speed and memory capacity.
6. Customer Specification and Reliability Requirements (Where and how used?)
   The level of requirements relating to testing, transportability, and end use of the product.
7. Development Environment (What complicating factors exist?)
   The relative impact of unique project conditions on the normal time required to complete the job, measured with respect to the organization, resources, program application and project size.

PRICE-S™ covers all of the factors that are entered into the COCOMO model. It is however, much more robust in its ability to evaluate each of these attributes in much finer detail. As an example, COCOMO does not consider the language except under the area of personnel experience. PRICE-S™ not only looks at personnel aspects but evaluates twenty-four different language types that have different developmental values (PRICE, 1989: 2-E-10). This fineness of detail applies to all attributes and conditions.

PRICE-S™ is one of the first commercial software tools that provides an expert interface. Its strengths lie in its consideration of multiple factors to a much greater degree than COCOMO and its inclusion of a software sizing and life cycle analysis capability. PRICE-S™ weaknesses are its difficulty to learn and to maintain proficiency. There is no PC version of PRICE-S™ which requires the user to have an expensive lease to a remote processing center. Lastly, it has proven to be inaccurate in ground based avionics applications (Ferens, 1990c).

System Evaluation and Estimation of Resources Model (SEER). This model is another proprietary commercial program similar in nature to PRICE-S™. This PC based software package is one of the most powerful software cost estimating systems on the market. It is designed to evaluate software cost for the entire life cycle of the system or it can be modified to provide costing for selected phases (McMurry and Nelson, 1990: 2).

SEER™ is unique in that it requires one hundred and fifty-seven inputs to perform its analysis. Its primary input categories identified on the SEER-SEM Early Estimate Worksheet (SEER™, 1988:iws-1n), are platform, application, development method, and development standards. These four categories are broken down further into forty-eight sub categories to provide the finest detail of the three tool. It elicits inputs for complexity, personnel capabilities, development support environment, product development requirements, reusability requirements, development environment complexity, target environment, schedule, staffing, probability, software requirements analysis, software to software integration, software to hardware integration, and software maintenance. It also has the ability to provide seventeen different reports with analysis of cost, schedule, detailed monthly expenditures and input relationships (SEER™, 1988:11-1).

SEER's™ strength lie in its many inputs and features. It uses program evaluation and resource tracking (PERT) methodology for all input, addresses different life cycle models, and complies closely with DoD-STD-2167A defense system software development terminology. Its weaknesses lie in its cost to lease, its difficulty to learn, and its complexity to run (Ferens, 1990c).

Expert System Development Shell

Expert system shells are the framework from which the expert system is built. They provide the integrated mechanism for managing the rules and facts of the knowledge base, the inference and control strategy, and the knowledge acquisition subsystem (Holt, 1990a).

Nexpert Object™. Nexpert Object™ is an object oriented expert system development tool. This tool creates an object base consisting of classes, subclasses, objects, subobjects, and properties. to represent the problem domain (Rasmus, 1989:145). If all aircraft were the class, then they would have wings, tails, and other structural items in common. Aircraft can be broken down into subclasses of fighters, bombers, and cargo, etc. A fighter can be classified as the object, F-16. The F-16 can be further broken down into subobjects, such as model types, with each of the models having properties that describe them down to the finest detail. During the inference process, the class and object property values are tied to control mechanisms called meta-slots (Rasmus, 1989:145). "The meta-slots determine how the property will affect or behave in relation to the inference mechanism as well as the inference display" (Nexpert Object™, 1989:G-11). In other words, the object A-7 Fighter Aircraft, would, when entered into the system, inherit all of the properties of the class 'aircraft', and the subobject 'fighters'.

There are basically three types of inference engines used in expert shells. Forward chaining inference engines evaluate data as it is entered. The engine searches for rules that the data satisfies and as the rules 'fire' new conclusions are arrived at until the system is driven to the final conclusion (Rasmus, 1989:139). Backward Chaining is an inference method where the system starts with what it wants to prove, e.g., Z, and tries to establish the facts it needs to prove Z (Waterman, 1986:77). Nexpert Object™ is a hybrid of the two systems which allows it to closer approximate the reasoning processes of the human expert (Rasmus, 1989:139). Nexpert's™ inference processor allows it to create dynamic objects by adding a DO command to its IF/Then rule structure. The dynamic object can inherit all of the attributes and properties of the classes and subobjects it is linked to.

The inference strategy that will be used for the development of this system will be an Unconditionally Forward Hypothesis that allows propagation of whatever value of the initial hypothesis. Forward action effects and exhaustive evaluation will be used to further refine the levels of agenda control. Forward action effects are invoked when the firing of a rule results in modification of the value of any element of the knowledge base (KB) . Exhaustive Evaluation is used because some hypotheses have more than one rule leading to them (such as the case when building two rules to build an OR operator)(Nexpert, 1989:Chap 4).

Using the above strategy for the development of the software cost estimation expert system, a way must be developed for creating classes of objects using software subsystems, modules, and modes. The purpose is to investigate the class modules, and pass on that information to the subsystem (inherit). Objects will be created within Nexpert™ to represent the individual subsystems and modules. They in turn will inherit the individual values from the users input and those created through data manipulation by the program (Nexpert, 1989:Chap 4). Objects can be created and inherit properties based on the class which holds the ideal object or the template for the system.

# III. Methodology

## Overview

This chapter discusses the methodology underlying the development of this thesis. Included are discussions on the methodology used, the system design, and the evaluation and validation of the conceptualized model.

## Research Methodology

The research methodology follows a knowledge engineering outline (Holt, 1990) which breaks the development of an expert system into four phases: Phase I is an attempt to solve the problem in a conventional manner without the aid of an expert; phase II entails translating the problem into meaningful statements or rules and includes interface with the system expert; phase III begins the translation of the expert's input into computer code; phase IV begins full operational use of the system. The following outline, states actions and questions for the entire methodological process.

### PHASE I

Attempt to solve the problem yourself - Investigate present software cost estimation procedures to include non-automated processes and computer assisted processes. Attention to the factors involved in making the cost estimation and to the validity of the models under analysis is necessary.

Gain experience with expert system development. Experiment with different types of expert shells with particular attention in comparing standard shells with object oriented shells.

Prepare questions for the experts. - Prepare a battery of questions for use in interrogating software cost estimation experts. The questions will be exploratory in nature that elicit a response other than yes or no. The questions will be general enough to focus the experts attention on the areas of concern, but structured to handle several levels of detail. The question's structure will provide enough flexibility for use in interrogating experts of different backgrounds and varying degrees of experience.

Question the experts. - Software estimating experts are required to provide common, verifiable, consistent, reliable answers to the questions needed by the users of this system. They will be interrogated with a series of questions that elicit responses necessary to aid the expert system user in focusing on the correct inputs for his problem. A questionnaire has been developed to ensure consistency of responses from one expert to the next (see Appendix C). The experts will be asked to provide data or advice they feels is salient to software cost analysis. A written set of the questions will be provided to the experts so they can refer to them as the interview progresses. The interview will be structured to handle several levels of detail with an emphasis on differentiating between important factors. If responses are made based on a position in a continuum, the experts will be asked how or why they arrived at that decision. The interview will be structured around an "If/Then" structure to allow for easier translation of the expert responses into ES rules.

PHASE II

Prepare translation of discussion to rules. Meaningful variables and common english terms will be used when possible. Rule networking will facilitate tracing of dependencies and inherited characteristics. Identify gaps, dead ends, holes. Rules will be cross referenced to the questions to ensure all variables identified by the expert are included.

Try to understand translation. Ensure translation maintains intent of the expert's input. An attempt will be made to understand the reasoning behind the expert's responses. Why did he say that? What are the driving factors? Think about gaps, dead ends, holes.

Forward rules to expert. Be prepared for negative feedback. Allow experts the opportunity to evaluate their responses to prevent inclusion of interrogator biases into the rule base.

Interview expert again. Re-evaluate rules with experts once all changes have been integrated into the expert system. Try to understand their new position if changes are made in their responses. Clarify gaps, dead ends, holes, and mis-rules.

PHASE III

Translate rules into computer code. Code the software to accomplish the software cost estimation evaluations. To be effective, the expert system will be coded to:

- Be flexible enough for use by both novices and experts alike.
- Be able to update/change files.
- Be able to provide on-line help when needed.
- Be capable of report generation.

The user interface will be coded to include as many of the guidelines derived from the interrogation of the expert and augmented with active cues for on screen help. Use expert system transcript to ensure code traceability and documentation of changes. Document mismatches and personal additions.

Enter computer code into system. Integrate rules and algorithms into expert system knowledge base. Document reasons for rules. Develop formal tree of relationships.

Expert System Design. The design of this system will be based on the integration of the software cost estimation algorithms identified in the detailed COCOMO model (Boehm, 1981:347), with expert insight into cost estimation attributes. The integrating tool for this effort is Nexpert Object™ expert system development shell. This shell, as discussed in chapter II, provides the knowledge engineer the ability to identify major classes that have particular attributes that can be passed on to other sub-classes. The purpose is to evaluate the class and pass on that information to the subsystem (inherit). Individual subsystems and modules will be created through the create object function within Nexpert (dynamic object) which will pass the inherited values of the individual attributes from the users input and that created through data manipulation by the program (Nexpert, 1989:Chap 4). Test system to ensure fundamental requirements are functional.

Test with expert. Allow the expert to exercise the system with variety of different cases. Try to break the system with borderline cases and then consider new rules to clarify and expand the system's capability. Make changes as required and integrate into knowledge base.

PHASE IV

Operate system. Develop system for the novice user. Interface will be developed to facilitate ease of use by personnel with limited experience. Test using actual field case proposals and compare to historical results. Document mismatches and mistakes for future correction and modification.

Test and Evaluation

The bottom line of this software development effort is proving that the system will do what it is designed to do. Boehm discusses software verification and validation (V&V) and depicts it as spanning the entire software development life-cycle (Boehm, 1981:36).

*Verification:* To establish the truth of correspondence between a software product and its specification (from the latin *veritas*, "truth").
*Validation:* To establish the fitness or worth of a software product for its operational mission (from the latin *valere*, "to be worth").

In other words is the expert system appropriate for the task it has been designed for and is it being built correctly?

To verify the effectiveness of the software cost estimating expert system being built, it must be compared to already existing models that have proven track records. Consequently, the two models discussed in chapter II, PRICE-S™ and SEER™, will be exercised using the same input variables and then comparisons made among them to ensure they are consistent in their output. To do this, a case study (Ferens, 1990) ,written for the IMGT 676, Software Cost Estimation Class (Appendix E), will be used to provide a baseline of information the models will be tasked to evaluate. Once all the data has been evaluated by the standard parametric models, the data will be analyzed by the expert system.

The above process verifies the ability of ES to accomplish the mathematical operations required to provide the output appropriate to the tasking. There is, however, one other test that must occur before the new model is completely verified. The model must be used by a variety of users, with different capabilities, to verify that it does indeed provide expert advice and then give consistent results.

The expert system model will be provided to engineers, cost analysts, and middle managers to see if it facilitates the analysis of software proposals and provide predictable results as a function of that aiding. In order to do this, actual software acquisition case files will be used to exercise the system and the users. The results of the prior runs and the ES run will be compared to see if there are any discrepancies between the case file results and those provided by the ES. Evaluations will be made to ensure discrepancies identified belong in one of two categories, expert system or human error. Identified faults in the software they will be repaired and the data reentered. If human error is determined, then both the person and the software will be evaluated to ascertain if the software could have prevented the error by providing better expert advice.

A realistic goal for any software system is the practicality or benefit the user can derive out of its use. Boehm identified a series of questions that are a test of the practicality of software systems. Each of the models used in this effort will be compared to see how the new expert system fares against this criteria.

1. Definition - Has the model clearly defined the costs it is estimating?

2. Fidelity - Are the estimates close to the actual costs expended?

3. Objectivity - Does the model avoid allocating most of the cost to subjective factors?

4. Constructiveness - Can the user tell why the model gives the estimate it does?

5. Detail - Does the model consider the number of subsystems, units, and phases?

6. Stability - Do small differences in input provide small differences in output?

7. Scope - Does the model cover all classes of software you wish to estimate?

8. Ease of use - Are the inputs and options easy to understand?

9. Prospectiveness - Does the model avoid the use of unknown data?

10. Parsimony - Does the model avoid redundant or unnecessary factors?

(Boehm, 1981:476)

# IV. Results and Findings

## Overview

This chapter presents the results of the research, an analysis of the expert interviews, a comparison of the models used, and a discussion of the findings.

## PHASE I - Attempt to solve the problem yourself

Non-Automated Methods. To solve problems in software cost estimation, one must look at the mechanisms presently used today. Automated models, such as PRICE-S and SEER, which will be discussed in following paragraphs, use proprietary rules and algorithms that have been developed and proven over time. Independent of these models are the non-proprietary algorithms that define the computational structure of the COCOMO model (see Appendix E).

Boehm developed a Software Hierarchy Estimating Form (SHEF) to facilitate computation of software cost (Boehm, 1981:348-353). It is necessary when using this form to go through a rigorous twenty-five step procedure to compute eve simple software cost problems. Knowledge of the formulas identified in Appendix E was essential as was an understanding of the qualitative descriptions of each of the attributes. To compute the solution, the evaluator must interpret the software proposal and evaluate each of the fifteen attributes relative to the description provided. He must designate the stage of development, specify if the factor is at the subsystem or module level, and then assign a rating predicated on this subjective evaluation. Each of the attributes has its own continuum which the user must be experienced with to assign the appropriate rating. An example of one of the fifteen rating scales is given in Table 5.

Table 5

Required Reliability (RELY) Rating Scale
(Boehm, 1981: 374)

| | |
|---|---|
| Very Low | The effect of software failure is simply the inconvenience incumbent on the developers to fix the fault. |
| Low | The effect of software failure is a low level, easily recoverable loss to the users. |
| Nominal | The effect of a software failure is a moderate loss to the users, but a situation which one can recover without extreme penalty. |
| High | The effect of a software failure can be a major financial loss or a massive human inconvenience. |
| Very High | The effect of a software failure can be loss of human life. |

Each of the fifteen attributes has a table similar to this that the evaluator must interpret. The more experience he has in doing this, obviously the better. Once a rating and the phase of development has been identified, the actual quantified rating is assigned and integrated into the algorithms so the SHEF computations can be performed. Some of the tables, like the execution time (TIME) constraint, are easy to evaluate and assign a rating. Others, like system complexity (CPLX), are much more difficult to understand and consequently more difficult to assign an accurate rating.

Boehm provides an example problem for students to test their skills in computing software cost (Boehm, 1981: 341-352). The system under evaluation consisted of three subsystems made of six modules total. Even though this is a relatively simple problem, the computational requirements were extremely high. Computations at the module level were done first and required the assignment of 56 ratings similar to those found in Table 5. Once those values were assigned, no less than 151 computations had to be done to complete the form at the module level. The next step was to complete the analysis at the subsystem level which required 60 lookups for rating assignment and 91 mathematical operations to complete the form. The grand total for this effort was 116 lookups from tables and 241 mathematical operations. Needless to say the potential for error was great.

In the final analysis, hand computation using the SHEF form or any other non-automated spreadsheet is cumbersome at best. This analysis places a good deal of responsibility on the user in understanding the terms, manipulation of the algorithms, and selecting the appropriate ratings for the attributes. However, because it is spreadsheet in design one could easily build a Lotus type template to accomplish the task. It would not solve the major problem though of providing insight to the user of how to assign the ratings.

Automated Methods. The two automated software cost estimating models identified in Chapter II, PRICE-S and SEER, were evaluated using the Cost 676 case study (Appendix D). The case study was used to provide a baseline for comparison of the above models by ensuring each was tested using the same environmental, personnel, project, and computer attributes. Consistent among both models was their overall difficulty to use and their complexity of design (Goodson and Pshsynchyniak, 1990:13; McMurry and Nelson, 1990:9). A comparison and evaluation of the practicality of these models is made at the end of this chapter.

The overall benefit of models of this type are the consistency of results for similar systems evaluated and the fine level of detail they evaluate to ensure as accurate a computation as possible. A comparison of the results of PRICE-S™ and SEER™ is provided below in Table 6. A significant factor to note is the significant difference between the two models in their cost estimations. Mr. Dan Ferens offered the following insight that not all models evaluate cost in the same fashion. The primary inputs for PRICE-S is man months and the primary for SEER is source lines of code. The important issue is that your model is consistent when compared to historical data from your particular program.

Table 6

PRICE-S Verses SEER
(Goodson and Pshsynchyniak, 1990:8;
McMurry and Nelson, 1990:13)

| | Arcturus | Vega | Pollux |
|---|---|---|---|
| PRICE-S | | | |
| Development | $35.6 | $47.5 | $57.8 |
| Support | 26.7 | 25.8 | 45.1 |
| Total | 62.3 | 73.3 | 102.9 |
| | | | |
| SEER | | | |
| Development | $.104 | $.097 | $.091 |
| Support | .283 | .230 | .451 |
| Total | .387 | .327 | .542 |

Cost in Millions

Additionally, as more data is gathered and the data base expanded, the easier it will be to calibrate the models to fit specific applications or contractors. Calibration will tend to bring both models closer to similar answers.

Expert system shell selection. The original criteria for ES shell selection was that the system be compatible with a Macintosh computer and that it have a certain degree of transportability to other systems. Nexpert Object™ was available and fit the above requirements.

Question Preparation. A series of questions were developed (Appendix C) that reflected the basic design structure of the Detailed COCOMO cost estimation model. The questions were structured in a way to handle as much detail about the problem as possible, but made generic enough as to not lead the interviewee to a specific answer. Additionally, there were questions aside from those specific to COCOMO included to ensure that any important variables were not overlooked.

Questioning the expert. Each of the interviewee's was provided a set of the questions to ensure he could follow the interview without distraction. Having the questions in hand, in addition to be asked, allowed the expert to better internalize the problem area and reflect on alternative solutions. Due to problems in coordinating schedules with the experts they were also given the questions ahead of time which allowed them the luxury of reviewing them before the interview.

One outcome to the interrogation was the experts perceiving their problem domain as essentially "not expert". Their expertise was applied in the analyses of the contractors proposal and not in the determination of software cost effort multipliers. This position however inconsistent with what was expected, still provided expert insight into the problem domain but from another equally viable perspective. If difficulties arose in determination of the cost factors, they would simply look in the users manual for assistance in making that decision. Indeed, if the problem was more difficult than the manual could provide an answer for, they would call the support contractor, for that particular model, and get a best estimate from them.

The end result of the interviews was limited insight into the problem domain other than using the manuals and books as guidance in determining cost estimation attributes. The individuals expert insight, compounded with the data bases and studies in Boehm's book provide sufficient expert knowledge though to build the expert system.

PHASE II - Prepare translation of discussion to rules.

Probably the most challenging aspect of this expert system was creation of the rules and knowledge base from information provided in the expert analysis. The rules were first written in a non-machine pseudo code which allowed for a general mapping and understanding of the problem domain. During the development of the preliminary code it was necessary to discuss gain an understanding of how the ES shell would manipulate the information in order to arrive at the predefined goals. Nexpert™ required an understanding of object oriented programming, that without formal prior knowledge, was difficult to understand. The shell's architecture provided a good explanation facility that allowed programming to begin without much delay.

Once the pseudo code was complete, conversion to Nexpert™ object oriented code was made. Even though the syntax of the code was somewhat esoteric, every attempt was made to make the variables as meaningful as possible. By doing this it allowed for easier tracing for fault detection and debugging. The following rule is an example of the rules developed for this system (see figure 3). The purpose of this rule is to assign values which will be used in computing the phase development schedule. In this case the user provides an input in response to a computer query on whether the subsystem development schedule is 'stretched out' or 'normal'. If it is stretched out then the class subsystem with the property of schedule is set to high. Each of the values is assigned to the different algorithms used to compute the individual phase schedules. In this rule the class |subsystem|

has a property called Sced or schedule. The first sub-class, <|Subsystem|>, has a property called Scedpd which stands for the schedule for preliminary development. Once the action is confirmed, the value of 1.1 is assigned to the algorithm containing this variable. All objects that use this variable will inherit that value from this one action. Each of the other values are similarly assigned to properties. See Appendix G for a complete listing of the rule base.

If
|SUBSYSTEM|.Sced is "Stretched out development schedule"
Then **Action 100**
is confirmed
And 1.1 is assigned to <|Subsystem|>.Scedpd
And 1.1 is assigned to <|Subsystem|>.Sceddd
And 1.0 is assigned to <|Subsystem|>.Scedcut
And 1.0 is assigned to <|Subsystem|>.Scedit
And <|Subsystem|>.Sced rating is set to "High"

Figure 3. Nexpert™ Object Oriented Rule

PHASE III - Translate rules into computer code.

One of the convenient advantages of Nexpert ™ is once you have written your rules, you have basically written your computer code. There is no requirement to convert the rules to a code of any lesser degree than the object oriented code which is an inherent part of Nexpert's™ structure. To ensure complete traceability of the designed system, Nexpert™ provided the capability of reviewing the rules base, the knowledge base, the transcript, and all objects with their classes and properties. A complete object network allowed for a visual representation of the system as a final check of the interrelationships of the objects, their properties and the hypothesis that control them.

The inference strategy used for the development of this system an unconditionally forward hypothesis that allowed propagation of whatever value of the initial hypothesis. Forward action effects and exhaustive evaluation were also used to further refine the levels of agenda control. Forward action effects are invoked when the firing of a rule resulted in modification of the value of any element of the knowledge base (KB) such as shown in table 7. Exhaustive evaluation was used because some hypotheses had more than one rule leading to them.

During the development of this system the ES shell exhibited a large number of faults or bugs. Because of this an upgrade was made to the software that allowed for greater fault isolation and improved user interface. The negative part of this upgrade was to make the previously developed software cost estimating system partially incompatible with the new software. Because of the lack of time to resurrect the software, tests with software cost analysts were not possible.

PHASE IV - Operate system.

An attempt was made to reconfigure the ES to the new software configuration in order to have novice cost estimators test the system. To effect that goal a multilayered user interface was to be integrated into the system using the standard Hypercard™ information organizer. However, due to the software upgrade problems cited in Phase III and the lack of time to recode the system, this was not possible.

Despite the problems with the software there is little doubt that given time, a viable interface and powerful software cost estimating tool could be developed. As the system was exercised using program office case files, it could be calibrated to the programmatic environment of the system program offices at ASD.

Model Comparison

Boehm identifies ten criteria (Boehm, 1981:520-523) for evaluating a software cost model for practical estimation utility. Each of the models used in this study have been evaluated against this criteria in order to obtain an understanding of the usefulness of the expert system. The ratings for PRICE-S are synthesized from the study done by Goodson and Fshsynchyniak while the ratings for SEER where derived from the McMurry and Nelson report. See Table 8.

Table 7
Comparison of Parametric and Expert System Models

Definition - Has the model clearly defined the costs it is estimating?

| | |
|---|---|
| PRICE-S | Good |
| SEER | Good |
| ES | Good |

Fidelity - Are the estimates close to the actual costs expended?

| | |
|---|---|
| PRICE-S | Good |
| SEER | Fair |
| ES | TBD |

Objectivity - Does the model avoid allocating most of the cost to subjective factors?

| | |
|---|---|
| PRICE-S | Good |
| SEER | Fair |
| ES | Poor |

Constructiveness - Can the user tell why the model gives the estimate it does?

| | |
|---|---|
| PRICE-S | Fair |
| SEER | Fair |
| ES | Good |

Detail - Does the model consider the number of subsystems, units, and phases?

| | |
|---|---|
| PRICE-S | Good |
| SEER | Good |
| ES | Fair |

Stability - Do small differences in input provide small differences in output?

| | |
|---|---|
| PRICE-S | Good |
| SEER | Good |
| ES | Fair |

Table 7
Comparison of Parametric and Expert System Models
(continued)

Scope - Does the model cover all classes of software you wish to estimate?

| | |
|---|---|
| PRICE-S | Fair |
| SEER | Fair |
| ES | Fair |

Ease of use - Are the inputs and options easy to understand?

| | |
|---|---|
| PRICE-S | Poor |
| SEER | Poor |
| ES | Good |

Prospectiveness - Does the model avoid the use of unknown data?

| | |
|---|---|
| PRICE-S | Good |
| SEER | Good |
| ES | Good |

Parsimony - Does the model avoid redundant or unnecessary factors?

| | |
|---|---|
| PRICE-S | Poor |
| SEER | Poor |
| ES | Good |

To gain a numerical comparison of the models, each of the ratings was assigned a value. Good was rated as a three, fair rated a two, and poor rated one point. When these values are tallied PRICE-S rates the highest with twenty-four total points or an average of 2.4. Even though the expert system received no points for Fidelity due to lack of data, it tied in overall points with SEER™ with a rating of 22 and an average of 2.2. This rating tends to show that the ES, despite the design weaknesses identified earlier, has the potential for being competitive with the larger more expensive systems.

# V. Conclusions, and Recommendations

## OVERVIEW

This chapter summarizes the research objective, methodology, and findings, emphasizing conclusions about the effort, making recommendations for further use of methodology, and suggesting further research.

### Summary

The purpose of this study was to develop a software cost estimating expert system. A series of questions were given to software development personnel who have exhibited expertise in ascertaining software development cost. Their answers were evaluated and transformed into an expert system capable of aiding cost analysts of differing degrees of experience.

This study met with limited success. There was a general recognition that experts do use expert knowledge in the evaluation of software development proposals. They tend, however, to rely strongly on the software cost estimation models to make their decisions as opposed to adding their own expertise.

### Conclusions

The following paragraphs provide a discussion on the study results relative to the research questions.

Question 1: What knowledge and heuristics, rules of thumb, do software cost analysts use in decision making? The expert's knowledge about the problem domain is derived from formal training in cost estimation techniques and from their experiences in Air Force cost analysis centers. The knowledge is focused on interpreting software development proposals and on the manipulation of a variety of parametric cost estimation tools. They augmented their expert knowledge by using the cost model manuals and directives, like DoD-Std-2167A. The transformation of that knowledge into values or data that could be used in a standard parametric models was further aided by the model's basic menu driven help screens and contractor assistance when required. The experts felt their experience tied with the integration of guidance from many sources allowed them to effectively analyze the contractor's proposal to ascertain the appropriate inputs for the models.

One heuristic used consistently by the experts was to rate all software systems as 'embedded'. Other heuristics were identified as situationally dependent and difficult to qualify during the expert interviews.

Question 2: Can this knowledge and these heuristics can be captured and programmed as an "expert system?" The insight and knowledge from the experts was valuable in selecting the appropriate rules and algorithms for building the expert system. The information of greatest importance in programming the ES was how to ascertain the appropriate rating for a given software cost estimation attribute. This information allowed for the ES to be programmed to a level that novice cost estimators could use the system and not have to go to other sources for clarification or guidance.

Question3: Do expert system significantly improve the performance of software cost analysts? This question was not satisfactorily answered by this research. The general feeling among the software cost experts who were interviewed was that it would have a significant impact on improving performance. Their position was based on the feeling that too much time was spent by cost analysts in figuring out the eccentricities of the different estimation models. The use of an effective interface to provide the guidance and insight to the user would allow them to spend more time in evaluating the proposals and less on interpreting the demands of the cost tools. The consensus was that expert systems would be beneficial in the evaluation of proposals and enhance the cost analyst's capability to more accurately estimate program development, support, and maintenance costs and thereby better program for future budgetary requirements.

Question 4: Can the use of expert systems allow cost analysts to solve complex or difficult problems more efficiently? One of the major concerns of the experts who were interviewed was the difficulty in using the present methods of cost estimation. This concern is confirmed in this study in the comparison of the two parametric models and how difficult it was to use them for even simple problems. This study did not, however, show that the ES could improve efficiency given a complex cost estimation problem. Similar to question three above, again it was felt that this type of system could improve efficiency by providing the cost estimation decision maker with the tool and information necessary to make sound decisions.

## Recommendations for Redesign

The following measures of system effectiveness offer the potential for further research and system redesign.

- Productivity measure. Presently there is no mechanism built into the system to capture the results of past decisions and provide them to the user as a guide for future decision making (calibration of the algorithms).

- Process Measure. The system requires better integration of the analytical tools and needs to expand on its alternative generation capability which will provide a better problem analysis.

- Perception Measure. Needs better support of the judgments the user must make. Additionally, within the scrollable screens there is no aiding to the user on how to assess the value of certain parameters. The reasoning process is also poorly supported which may lead the user to wonder on the decisions validity.

In addition to the above items, the following would be beneficial to users of all experience levels in improving system utility.

- Create a running log or transcript that the user can dump to the screen or printer so he can audit his actions if required.

- Provide "hot spots" in the scrollable fields that will take the user to supporting fields, such as graphs, charts, and tables.

- Create a model dictionary that will allow the user to bring in array of different models or parameter changes to existing models to make better decisions.

## Further Research

Managers are not lacking in the availability of data or models to manipulate their data. What they do lack is a way to integrate the data and the models into a coherent system in support of decision making.

### Software Cost Estimation Decision Support System

A DSS, unlike the expert system created in this study, would have to integrate not just one model, but two or more. For this effort there are three areas that require modeling. They are software cost estimation, contractor performance evaluation, and schedule development.

The cost estimation model must be capable of determining the capabilities of the contractor's software development team, hardware requirements in support of the software, and any constraints that are inherent in the design. Additionally, life cycle support and maintenance costs need to be determined and if possible, provide a schedule for development and delivery of the product.

The contractor performance model should have the capability of providing the decision maker aid in determining the contractor(s) technical performance relative to the descriptions provided in the contractor's proposal.

The schedule development model should have the ability to provide insight to the user for determination of the software development schedule. At a minimum it must include the DOD-STD-2167A development cycle and a schedule of all tests, design reviews, and audits.

## Appendix A: Definitions

Algorithm - A formal procedure guaranteed to produce correct or optimal solutions (Waterman, 1986:22).

Artificial Intelligence (AI) - The field of computer science concerned with developing computer systems that exhibit characteristics of human intelligence associated with tasks as learning, problem solving, pattern recognition, and reasoning (Young, 1989:343).

Backward Chaining - An inference method where the system starts with what it wants to prove, e.g., Z, and tries to establish the facts it needs to prove Z (Waterman, 1986:77).

Class - Collection of objects; functionally a generalization since the properties of the class are found in its instance or objects (Nexpert Object,™ 1989: G-2).

Create Object - Operator which allows rules, Order of sources, or If Change slots to create new object entities, and link them to one or several classes or to other dynamically created objects. The Create Object operator can also establish new links between existing objects and classes defined in the knowledge base (Nexpert Object™, 1989:G-3).

Domain - A topical area or region of knowledge (Stepanek,1988:D-2).

Dynamic Objects - Objects which are created either by rules, Meta-Slots or by an external call (Nexpert Object™, 1989:G-5).

Exhaustive Evaluation - Inference strategy setting indicating that all rules pointing to the current hypothesis must be evaluated (Nexpert Object™, 1989:G-5).

Exhaustive Search - A problem-solving technique in which the problem solver systematically tries all possible solutions in some "brute force" manner until it finds an acceptable one (Waterman, 1986:151).

Expert System (ES) - "a computer program using expert knowledge to attain high levels of performance in a narrow problem area" (Waterman, 1986:11). A heavily symbolic computer program which possesses a separate declarative knowledge base and inference engine, and which is generally implemented using ES tools, technology, and languages (Stepanek,1988:D-2).

Explanation Facility - That part of an expert system that explains how solutions were reached and justifies the steps used to reach them (Waterman, 1986:30).

Forward Confirmed Hypothesis - A strategy setting allowing context propagation if the source hypothesis is true (Nexpert Object™, 1989:G-6).

Heuristic - A "rule of thumb" which, while not specifically accurate, may narrow down the range of possible solutions to the problem (Stepanek,1988:D-3).

Inference Chain - The sequence of steps or rule applications used by a rule-based system to reach a conclusion (Waterman, 1986:78).

Inference Engine - That part of a knowledge-based system or expert system that contains the general problem-solving knowledge (Waterman, 1986:22).

Inheritance - The ability to transfer values and functions declared in an object or a class to other objects and/or classes related to them. The relationships are parent/child relationships. In Nexpert™, objects can inherit properties and their values as well as functions (methods) from their classes (which are more generalized structures). The reverse is also possible. Inheritance in Nexpert™ can be completely use-defined for every property of an object or a class using the Meta-Slot structures (Nexpert Object™, 1989:G-7).

Inheritance Hierarchy - A structure in a semantic net or frame system that permits items lower in the net to inherit properties from items higher in the net (Waterman, 1986:78).

Knowledge Base - The section of the ES code which contains the domain and problem solving knowledge (Stepanek,1988:D-3).

Metaknowledge - Knowledge in an expert system about how the system operates or reasons. More generally, knowledge about knowledge (Waterman, 1986:30).

Meta-Slots - System properties of a given property of either an object or a class. They determine how this property will affect or behave in relation to the inference mechanisms as well as the interface display (Nexpert Object™, 1989:G-11).

Object - This refers to physical or conceptual entities that have many attributes (Stepanek, 1988:D-4).

Parametric Model - A regression based model that relates estimated costs directly to parameters that describe the design, performance, schedule or operating environment to known or similar systems (Ferens, 1990c).

Rule - A formal way of specifying a recommendation, directive, or strategy, expressed as IF *premise* THEN *conclusion* or IF *condition* THEN *action* (Waterman, 1986:23).

Semantic Net - A knowledge representation method consisting of a network of nodes standing for concepts or objects connected by arcs describing the relations between nodes (Waterman, 1986:78).

Symbolic Reasoning - Problem solving based on the application of strategies and heuristics to manipulate symbols standing for problem concepts (Waterman, 1986:31).

# Appendix B: Selection Criteria for an Appropriate Domain for an Expert System

(AI Magazine, 1985:26-30)

Prerau's criteria as quoted from The AI Magazine, are in quotes, the effect they have on software cost estimation follow.

## BASIC REQUIREMENTS

- "The domain is characterized by the use of expert knowledge, judgment, and experience." The purpose is to use experiential information derived from an expert in software economics and apply to this program.

- "Conventional programming (algorithmic) approaches to the task are not satisfactory." "If a conventional approach will work well, there is usually less technical risk to using it rather than an expert system approach (Allen, 1989)." In the case of software cost estimating, there are many spreadsheet type programs available to do the nuts and bolts type of number manipulation. They do not, however, provide adequate insight to the novice cost estimator on how to weigh the factors and parameters necessary for optimum analysis of the data.

- "There are recognized Experts that solve the problem today." Experts do exist in the Air Force accounting and contracting community who analyze data of this type (Brill, 1989). Historically though, the AFSC product divisions utilize support contractors who are used to do software cost estimating.

- "Expertise is not or will not be available on a reliable and continuing basis, i.e., there is a need to capture the expertise." Given the major cuts in today's budget there has been a drive to reduce reliance on support contractors and to reduce manpower overall throughout the Department of Defense. By capturing the expertise required we are ensuring the capability of program managers to accurately forecast software cost with the use of an expert systems and personnel who have limited experience in software cost estimation.

- "The complete system is expected to have a significant payoff for the corporation." Payoffs can be realized by reductions in manpower, reduced times in negotiating contracts, and better discrimination of cost overruns.

- "Among possible application domains, the domain selected is that one that best meets overall project goals regarding project payoff versus risk of failure." The reaching of a satisfactory solution for software cost estimating must be balanced equally with the risk of failing to do so and effecting program schedules.

TYPE OF PROBLEM

- "The task primarily requires symbolic reasoning." There is little doubt that the basis of software cost estimating is uantifiably based. The requirement is to provide the cost estimator with discrete reasoning prompts that will elicit a response similar to one provided by an expert.

- "The task requires the use of heuristics, e.g., rules of thumb, strategies, etc. It may require consideration of an extremely large number of possibilities or it may require decisions to be based upon incomplete or uncertain information." The application of heuristics is essential in that we can begin to bring in the human elements of guessing, testing for possibilities, and entering into the equation what is commonly called "a gut feeling."

- "The system development has as its goal either to develop a system for actual use or to make major advances in the state of the art of expert system technology, but does not attempt to achieve both of these goals simultaneously." Ideally, once this system is fielded and tested, it will become a viable tool for anyone wishing to apply it in the software cost estimating domain.

- "The task is defined very clearly: At the project outset, there should be a precise definition of the inputs and outputs of the system to be developed." This part of the process is evolutionary in nature and should grow as the system develops (Allen, 1989:207). As the system manifests itself and is tested, or the environment changes, or any other intervening causes a change in development, the task parameters must be flexible enough to ensure ease of maintenance of the program.

THE EXPERT

- "There exists an expert to work with the project." It is these experts who will provide the heuristics, the decision rules, and the correct analysis of the program outcomes.

- "The expert's knowledge and reputation must be such that if the expert system is able to capture a portion of the expert's expertise, the system's output will have credibility and authority." New technology, of which expert systems are a member, suffer the bane of inertia. People dislike new, better ways of doing things, consequently the resistance to change can be great. An expert developer who has already established himself in this arena can provide legitimacy and credibility to the system even before it is tested and used.

- "The expert has built up expertise over a long period of task performance." The more experience our expert has the greater the probability of success for a performing expert system. This is a result of having the "feel" for how things should be done along with the practice of having already done the mechanics.

- "The expert will commit a substantial amount of time to the development of the system." Getting a supervisor to release their expert for extended periods of time can indeed be one of the major obstacles in the development of this system.

- "The expert is capable of communicating his knowledge, judgment, and experience, and the methods used to apply them to the e particular task."

- "The expert is cooperative." Willingness to work in a group think environment and participating in the success of the program makes for a better development environment for all concerned.

- "The expert should be easy to work with."

- "The expertise for the system, at least that pertaining to one particular sub-domain, is to be obtained primarily from one expert." It is seldom that experts can consistently agree on how to do any process. Therefor, by having one expert we reduce our chances of argument and indecision due to too many correct answers.

- "If multiple experts contribute in a particular sub-domain, one of them should be the primary expert with final authority." One person must break the ties or make the ultimate decision if conflicts arise.

PROBLEM BOUNDS

- "The problem is neither too easy (taking a human expert less than a few minutes) not too difficult (requiring more than a few hours for an expert)." Expert system takes a great deal of time and effort to do properly. If simpler methods can indeed satisfy the requirements then they should be used.

- "The amount of knowledge required by the task is large enough to make the knowledge base developed interesting."

- "The task is sufficiently narrow and self-contained: the aim is not for a system that is expert in an entire domain, but for a system that is an expert in a limited task within the domain." By limiting this project to software cost estimating , we have precluded the rest of software costing economics which is a broader field by far. This scoping is critical in making the finding of a system expert realistic and also makes it doable in the time allotted.

- "The number of important concepts (e.g., rules) required is bound to several hundreds."

DOMAIN AREA PERSONNEL

- "Personnel in the domain area are realistic, understanding the potential of an expert system for their domain, but also realizing that thus far few expert programs have resulted in actual production programs with major industrial payoff." Expert system development is a relatively new technology with only a few success stories. As a consequence then, there is potential for frustration and lost motivation due slow or no progress.

- "Domain area personnel understand that even a successful system will likely be limited in scope and, like a human expert, may not produce optimal or correct results 100% of the time."

- "There is strong managerial support from the domain area, especially regarding the large commitment of time by the expert(s), and their possible travel or temporary relocation, if required." Experts do things (like their assigned jobs) other than developing expert systems. For a supervisor to give up a major contributor to another agency is in most cases a great sacrifice. Having managements support in this issue can only aid when there is a conflict of priorities.

- "The specific task within the domain is jointly agreed upon by the system developers and the domain area personnel." This will ensure that the system the user requires will in fact be built by the system developers.

- "Managers in the domain area have previously identified the need to solve the problem which the system attacks." If management sees a need for the system then development will be supported.

- "The project is strongly supported by a senior manager, for protection and follow-up." There are times in the development of any system, whether it be hardware or software, that not everything goes according to plan. Budget cuts, technology changes, and program slips are just a few problems that need the intervention of senior management to keep the program on course and viable.

- "Potential users would welcome the completed system." The reason for this development in the first place is to satisfy a user's need.

- "The system can be introduced with minimal disturbance of the current practice." The purpose of this system is to enhance the users capabilities. By making it easy to learn and easy to integrate into the work centers, greatly increases the potential for system acceptance with the least disturbance.

- "The user group is cooperative and patient." Cooperative in that they provide information and advice when requested and patient when development falls behind schedule or runs into problems.

- "The introduction of the system will not be politically sensitive or controversial."

- "The knowledge contained in the system will not be politically sensitive or controversial."

- "The systems' results will not be politically sensitive, or controversial."

<u>OTHER DESIRABLE FEATURES</u>

- The system can be phased into use gracefully: Some percentage of incomplete coverage can be tolerated (at least initially), and the determination of whether a sub-problem is covered by the present system is not difficult." This feature allows for the early integration and test of the cost estimating model given that it will run in a modular or less that full-up form.

- "The task is decomposable, allowing relatively rapid prototyping for a closed small subset of the complete task; and then slow expansion to the complete task." Modularity allows for the ease of independent development of sub-domain elements, the upgrade of modules independently, and ease of maintenance as system requirements change.

- "The task is not all-or-nothing: Some percentage of incorrect or non-optimal results can be tolerated." In most cases optimality is not the key. Experts in a given field may not always provide a high percentage correct answers, so it is too with expert systems.

- "The skill required by the task is taught to novices." If the system can be taught to a novice it has structure and cognitive processes built within it.

- "There are books or other written materials discussing the domain." This information provides a boundary for the domain. It becomes evident that if the material is written it must have been written by someone who at least portends to be an expert.

- "The task's payoff is measurable." How else could the utility of the program be measured?

- "Experts would agree on whether the systems' results are good (correct)." Consensus is important in gaining acceptance of the program. The more experts agreeing the system will accomplish its designed task, the faster it will be accepted.

- "Test cases are available." Once the system has been fielded it must be tested. The best test of the model then is to use historical software cost estimating data and test the expert system for correlation to the aforementioned results.

- "The need for the task is projected to continue for several years." This allows us to monitor the efficacy of the expert system over an extended life cycle of software development. It also allows us to modify and hone the system to fit the specific requirements of each program it is being utilized on.

- "The domain is fairly stable. Expected changes are such that they utilize the strengths of expert systems (e.g., ease of updating or revising specific rules in a knowledge base), but will not require major changes in reasoning processes." Modularity of the expert system software design will ensure ease of update and change as the environment dictates.

- "The effects of corporate developments that will significantly hang the definition of the task can be foreseen and taken into account."

- "No alternative solution to the problem is being pursued or is expected to be pursued." Given the volatility of this technology, there is little doubt that alternatives will develop. If they turn out to be a better estimator then so much the better for the end user who is looking for improving their cost estimating capabilities.

- "The project is not on the critical path for any other development, and has not absolute milestones for completion." This project is on a critical path but for its own development. (See attachment 1, for critical milestones).

- "At the outset of the project, the expert is able to specify many of the important concepts." This provides us with a higher probability of developing a good task statement from the very beginning and not doing it piecemeal as the system evolves.

- "The tasks is similar to that of a successful existing expert system." The closer this expert system comes to a proven, successful system the better it will be in the long run. The gain from this is shortened development time and a good breakdown of lessons learned.

- "Any requirement for real-time response will not involve extensive effort."

- "The user interface will not require extensive effort." The key to the acceptance of any software based system is user friendliness. Is the model easy to use? Is it so complex that no one wishes to use it? Facility and speed are major factors in the acceptance of new technology developments.

## Appendix C: Expert Application Questionnaire for Software Cost Estimation

This questionnaire has been written to obtain expert knowledge on software cost estimation. In particular, the questions will conform to the parameters identified in the Detailed COCOMO software cost estimation model with additional queries for areas not covered by COCOMO. Those areas not covered by COCOMO will be highlighted with boldface type. In almost all cases the answering of the question will be followed by a "why" question, in order to clarify any misunderstandings that may occur.

Q1: How are you presently estimating software development costs?

Q2: Do you feel there are some models that do a better job than others?

Q3: What do you consider as the core requirements for estimating software cost? (ie, SLOC, function points, etc.)

Q4: Are there any algorithms or heuristics that you use or recommend, for ascertaining development cost?

Q5: Are there any non-cost variables to consider when evaluating a software proposal?

Q6 What is the impact of the above considerations on cost, if any?

Q7: How do you decide which Mode the software should be characterized as?

COST DRIVERS

Q8: What variables do you consider when evaluating a proposal for Personnel attributes.

Q9: What indicators do you use to rate the Analysts capabilities (ACAP)?

Q10: What indicators do you use to rate the development team's Application experience (AEXP)?

Q11: What indicators do you use to rate the Programmer capabilities (PCAP)?

Q12: What indicators do you use to rate the development team's Virtual machine experience (VEXP)?

Q13: What indicators do you use to rate the teams's experience with the programming language (LEXP)?

Q14: Are there any other personnel attributes you consider as important in determining software cost?

Q15: What variables do you consider when evaluating a proposal for Project attributes?

Q16: How do you rate the use of modern programming practices (MODP)?

Q17: What ratings do you use when evaluating the proposal for the use of software tools (TOOL)?

Q18: How do you rate the impact of schedule constraints on software development cost (SCED)?

Q19: How is managerial risk rated (**RISK**)?

Q20: How do you rate the impact of classified development (**SECU**)?

Q21: What variables do you consider when evaluating requirements volatility (**RVOL**)?

Q22: Are there any other project attributes you consider important in determining software cost?

Q23: What variables do you consider when evaluating a proposal for product attributes?

Q24: How do you rate the impact of software reliability (RELY)?

Q25: What do you consider when rating the data base size (DATA)?

Q26: What variables are considered when rating required reusability (**RUSE**)?

Q27: Are there any other product attributes you consider important in determining software cost?

Q28: What variables do you consider when evaluating a proposal for Computer attributes?

Q29: What criteria do you use when rating computer time constraints (TIME)?

Q30 How do you measure the impact of system storage (STOR) limitations on development cost?

Q31: What measures do you use when rating the virtual machine volatility (VIRT)?

Q32: How do you rate the development center's ability to turnaround output (TURN)?

Q33: Are there any other computer attributes you consider important in determining software cost?

OTHER COST DRIVERS

Q34: How do you compute the development schedule for each effort?

Q35: What considerations do you make when calibrating your model?

Q36: How do you rate the proposal for software support or maintenance costs?

Q37: Does the use of ADA have an impact on your determination of cost?

Q38: Are there any special cases, such as the development of an expert system, that would change any of the above variables?

Q39: How do you determine total life-cycle cost?

# Appendix D: Software Cost Exercise for COST 676: Software Cost Estimation

(Ferens, 1990b)

## PART I: DEVELOPMENT COSTS

### A. General Description

The Air Force C-28 System Program Office (SPO) at Aeronautical Systems Division (ASD) wants to acquire a new controls and displays program for the C-28 cargo aircraft, which is now in initial production and has completed all system-level reviews and audits. This avionics operational flight program will be built to military specification requirements and will be totally written in the Ada programming language (although certain portions may include 20% microcode). The C-28 controls and displays (C&D) program will be developed as a single Computer Software Configuration Item (CSCI) using the traditional "waterfall" approach to software development. Tailored editions of DoD-STD-2167A and associated data items are required. The program is scheduled to begin on 1 July 1990; no end date has been established, although support is scheduled to begin on 1 January 1998. Software engineering personnel in the C-28 SPO have estimated the size of the CSCI to be between 100,000 and 130,000 lines of Ada code, of which 30% may be type definition and data statements.

Three companies have bid on the contract: Arcturus, Pollux, and Vega. Best and final offers have been received, and no further discussions with the contractors are allowed. Earlier in the source selection, ASD C-28 SPO software engineers, along with a few software experts from the ASD engineering deputate, visited the three contractors' facilities to obtain information for contractor-peculiar model inputs (to be presented later). All contractors participated and were cooperative except Spica Technologies, Arcturus' subcontractor. (Arcturus provided the information for Spica.)

The bids for the CSCI development were as follows:

| | |
|---|---|
| Arcturus: | $9.1 million |
| Pollux: | $8.5 million |
| Vega: | $9.3 million |

These bids are in constant 1990 dollars, which are to be used in all cost estimates for development and support. No management reserve is allowed in the bidding or in cost estimates, although the contract is cost plus fixed fee. Procurement would like the C-28 SPO to select the lowest bidder, although they will consider any bid within 10% of the low bid if it can be sufficiently justified. Air Force Logistics Command (AFLC) has requested that predicted support costs, and supportability of the software in general, be considered in the selection of the development contractor. However, no support bids were required or submitted.

B. Project Description

The C-28 C&D CSCI is a highly interactive program with difficult response times and extensive interfaces to other C-28 system elements. Although the program is not "life-threatening" it is being developed to military specification requirements since a failure could cause high financial loss or may severely affect the mission. The data base size has a ratio of forty to sixty data base bytes per delivered source instruction, and the data base is complex in structure and interaction.

The complexity of the CSCI is judged to be "very high" with about fifty percent real-time code, advanced mathematics, and many difficult algorithms. However, use of the Ada language is expected to promote a well-structured program design with small modules. There will be moderate module coupling and cohesion. Despite the CSCI's nature as a "stand-alone" program, it does contain many interfaces and has special display requirements. The displays are interactive and controlled by the CSCI.

The host development system, similar for all companies, consists of a multiple VAX and a minimal Ada Program Set Environment (APSE). The target system is a single central processing unit distributed system, much like a communication's network, with a special operating system. The rehosting of the C&D CSCI from the host to the target system should not be difficult; it should only involve minor language and system differences. However, significant changes to the virtual machines of host and target systems may be expected every three months, and moderate changes may occur in the development practices and procedures throughout the project.

Reusability for this CSCI is not an issue because of it's unique application; and only small, non-critical changes are expected in the CSCI requirements during the project. Security is an issue, however, since the program is classified "Secret" and must conform to the "B-2" requirements of being segmented into protected critical and non-protected components and units.

Requirements definition for the CSCI is expected to be fairly clear, but difficult. Proposal-level requirements, which comprise about ten percent of the total requirements effort, will be completed at contract award. No formal requirements tools are mandatory.

During the effort it is expected that mature design concepts will be used in accordance with Ada and DoD-STD-2167A. However, some minor modifications due to technological advancement may occur, and Ada Programming Set Environment (APSE) development tools are expected to result in some productivity improvements. A "typical" integration and test percentage of 22% is expected, although quality assurance and test requirements must address the possibility of high financial loss. Testing is not too difficult, but will be extensive.

Development terminal response time is expected to be between a quarter of a second and a half of a second. Computer turnaround time is expected to be between six and thirty minutes. All companies will probably use four different work-station types.

All companies are assumed to have all staff members exempt from overtime pay, and should average 152 hours per month and 228 work days per year. Contractors will be expected to travel to the C-28 SPO and other agencies about four times a year.

## C. Company Description

1. Arcturus: Arcturus Software has been in business for twenty years and has about 1,000 people. They have developed many defense software programs as a subcontractor to several major aerospace companies. They have a very good record of meeting cost, schedule, and performance requirements. This is their first effort as a primary contractor. To perform part of the C-28 C&D CSCI effort, they have hired Spica Technologies as a subcontractor. Spica which has been in business six years and has about 200 people, is known for their high caliber personnel, and has developed several defense programs successfully when contracted alone. However, when they are subcontractors, they are often difficult to work with and are generally uncooperative with both the government managing agency and with the prime contractor. Arcturus is knowledgeable of the potential difficulties, but decided to "take a chance" because Spica has a display program that can be used in this effort.

For this effort, Arcturus will develop the controls program from scratch; the Ada lines of code are expected to be between 70,000 and 122,000; with 102,000 being "most-likely." The displays program will be developed by Spica. It is a 40,000 line of code program written in JOVIAL which must be converted to Ada. It is expected that 50% new design, 80% new code, and 30% new testing must be accomplished to recode this program and merge it with Arcturus' controls program.

The mix of code for the Arcturus CSCI is about 50% real-time with 70% new code; 10% on-line with 50% new design and all new code; 10% interactive with 50% design and all new code; and 30% mathematics with 70% new design and all new code.

The personnel for Arcturus and Spica are generally of high caliber. The analysts and programmers are mostly experienced and are better than 75% of their contemporaries. They have had three years' experience with similar programs, and about 2 1/2 years' experience with Ada programs. However, they only have about ten months' experience with the virtual machine.

Arcturus routinely uses modern design practices, and has extensive automated development tools, although there is little integration. They do possess a minimum APSE for Ada. They have about two years' experience with DoD-STD-2167A practices.

Arcturus is located in Dallas, TX, and Spica is located in San Antonio, TX, which is about 400 miles from Dallas. The developers must sometimes travel between the two locations for consultation, support, and resources.

Timing and storage constraints on the development and target systems are no problem for Arcturus' design; less than 50% of either are used. For Arcturus, this project may be described as a "mixture of new and repeated work", or a "normal new project". Both program design and user documentation have fully automated graphics and text support. There is no impact from commercial-off-the-shelf (COTS) software. Integration effort may be considered "closely coupled" using an experienced team, although there is a mix of experienced and new people.

Arcturus and Spica personnel are 100% dedicated to the project, and have good facilities with two people to an office. Their labor rate is $72 per hour, or $11,000 per month.

2. Pollux: Pollux Software Development, located in Orlando FL, has been in business for 10 years and has rapidly grown to 2,000 people. They have extensive experience with avionics software, including controls and displays programs. To bid on the C-28 C&D contract, they teamed with Castor

Research, a company who has some military experience including the development of a controls and displays program written in Ada for the A-30 aircraft. Castor has been in business for four years, and has about 150 full-time people working in Kissimmee, FL, about 30 miles from Orlando. Castor says they could modify their program very quickly and easily; but, in return, they would be allowed to retain all data rights to their component of the C-28 C&D CSCI. The result would be that only Pollux and Castor could support the program, and the cost would be 30% higher than "normal" due to sole-source support. AFLC does not like this idea at all, since they do not want to pay extra support money or be in a sole-source situation.

The teaming arrangement of Pollux and Castor is somewhat of an enigma. Pollux and Castor both have team chiefs who manage personnel from both companies. The Air Force expressed concern that Pollux may not be "in control", but Pollux and Castor both stated that it was "matrix management, like the Air Force uses". Travel between Orlando and Kissimmee will be frequent for consulting and resource usage.

The C-28 C&D CSCI design has Pollux developing the majority of code which is expected to be between 100,000 and 150,000 lines of Ada code, with 125,000 lines being "most likely". Castor's program contains 80,000 lines of Ada code of which 10% must be redesigned and retested and 20% recoded. This effectively results in a commercial off-the shelf (COTS) program with some impact on design and development due to interfacing requirements.

The CSCI consists of about 50% real-time code, 10% interactive code; 10% on-line code, and 30% mathematical code; all of which will require about 60% new design and 65% new code.

The personnel assigned to the Pollux-Castor team are top-notch. The analysts are very highly experienced and competent, and are in the top five percent of the field. Programmers are in the top quarter of their field. The team has had about six years' experience with similar applications, and about 2 1/2 years' experience with Ada. They routinely use modern design practices and have extensive automated development tools although there is little integration. The tools include a minimum APSE for Ada. They have two year's experience with DoD-STD-2167A practices, but only about ten months' experience with the virtual machine.

Timing and storage may be a problem due to the relatively large size of the CSCI. On both the development and target systems, timing and storage are expected to be about 70% of capacity. About 25% of the code is timeconstrained, and some overlaying or segmentation may be needed in primary storage.

Because Castor envisions sole-source support, they are not emphasizing documentation. Design documentation is partially automated, and user documentation has little or no automation for the programmers. The integration effort for the program is "closely coupled" using an extensively experienced team. The project is a mixture of new and repeated work, and may be considered a "normal new project" for the team.

The team's personnel are 100% dedicated to the project, and have adequate facilities with about three people per office. The team's labor rate is $92 per hour, or $14,000 per month. They attribute their relatively low cost to their cooperative teaming arrangement which avoids subcontracting fees and other management problems associated with subcontracting. If they are selected, they intend to form a separate company, Gemini Software Support Corporation, to perform support for the C-28 C&D CSCI.

3. Vega: Vega Software Works is a relatively small company of 500 people, located in Biloxi, Mississippi. They have been in business for about fifteen years, primarily in commercial software. They have written programs for banking and other financial institutions, and have been praised for their quality of work. Recently they have written some Ada programs for commercial use, and have begun incorporating fault-tolerant features in programs, especially exception handling, to circumvent errors. They have performed two small Ada software projects for military command and control applications, but have not yet implemented fault-tolerance in military software. The C-28 C&D CSCI will represent their first attempt at fault-tolerant military software. A single organization within Vega will be responsible for software development; no subcontractor or teaming arrangements will be used.

Vega plans to develop the CSCI from scratch, and expects it to contain between 90,000 and 132,000 lines of code, with 120,000 lines most likely". The personnel for Vega are generally competent; their analysts have a mixture of experience, and are about "average" compared with others. The programmers, however, are generally experienced and are in the "top quarter" of their field. The Vega team averages about 2 1/2 years' experience with Ada, but only has about ten months' experience with similar applications, the virtual machine, and DoD-STD-2167A development practices. Vega makes general use of modern design practices and has basic maxi-tools which include a minimal APSE.

For Vega's design, timing and storage constraints for the development and target systems are no problem; less than 50% of either is used. This project may be considered "novel", a new line of business for Vega. Their documentation capability is not extensive; they have partially-automated graphics text support for design documentation and little or no automation for user documentation. Vega, however, does intend to fully comply with the tailored DoD-STD-2167A documentation requirements for this project. Integration effort is "closely coupled" with a mix of personnel involved. There will be no impacts from COTS software since none will be used. The mix of instruction, all of which will be designed and coded from scratch, includes 50% real time codes 10% online, 10% interactive, and 30% mathematical. Vega's fault-tolerant design is expected to save 10% on support cost.

Vega personnel have about 60% access to resources, although the project will be fully-staffed. The office facilities in Biloxi are good, with two people sharing an office. Their labor rate is $72 per hour, or $11,000 per month.

## PART II: SUPPORT COSTS

### A General Description

AFLC plans to support the software at one of their five Air Logistics Centers (ALCs) for a 15-year period beginning 1 January 1998. The most likely choice is Warner-Robins ALC, where the C-28 aircraft is currently being supported; however, it is possible that Oklahoma City ALC may support it at a proposed consolidated flight program support center. Most support will be performed using the block change method with one block change expected per year.

AFLC cannot support the software with Air Force personnel because of insufficient availability. However, AFLC strongly desires to complete software support between the contractor chosen for development and Fomalhaut Software Company. (It should be noted that support can not be done by any company not selected for development; e.g. Vega can not support the software if Arcturus develops it.) AFLC currently favors Vega because their design is expected to reduce estimated support costs by 10%. They have expressed displeasure with Pollux, since they could not complete the effort and a 30% increase in support costs over estimates would probably result. They are slightly favorable about Arcturus, since they plan to support the software without subcontracting to Spica, and they can compete the effort with Fomalhaut.

### B. Project Description

During the 15 year support period, it is expected that 15% of the code will be changed each year, and that the software will grow by about 35% over the period. Enhancements and quality levels are expected to be "typical" for this type of project. The CSCI will be entirely supported at one location, however, there will be five operational C-28 locations which will submit problem reports, and may use slightly different configurations of the C&D CSCI.

The CSCI is expected to be reasonably up-to-date and be functionally partitioned to the lowest (unit) level. There are expected to be "average" hardware and software capacity limitations at the support facility. Keeping on schedule may be difficult, partly because the CSCI must interface with many other elements on the C-28 aircraft and the number of changes requested from the five operational locations.

It is not expected that the caliber of support personnel will change from that of the development personnel, even though Pollux and Castor expect to form the new Gemini Corporation, and Arcturus plans to drop subcontractor Spica. Also, no environmental differences are anticipated. There are a few differences among the three development contractors regarding software support, however. Because Pollux expects sole source support, they will have very little commented documentation and their listings will have little information. The documentation they will develop is expected to have a poor data dictionary. Arcturus and Vega are expected to have documentation with selected on-line comments, listings with extensive information and comments, and on "average" amount of documentation available.

COTS impact, varies among the three development contractors: Vega will have no COTS impact; Arcturus will have slight impact from the software developed by Spica, and Pollux will have much impact, because of Castor's proprietary program. Maintenance experience also varies. It is a "normal" activity for Vega and Arcturus, but is a "familiar" activity for Pollux.

C. Company Description: FOMALHAUT

Fomalhaut Software is a company of about 750 personnel located in Macon, GA near Warner Robins ALC. They have been in business about seven years They have extensive experience in software support, or "maintenance", since they have supported many programs of Warner Robins ALC.

Fomalhaut has about "average" programmers and analysts who have about 1 1/2 years' experience in Ada and 10 months' experience with the virtual machine. They have about six years' experience with similar applications, but only about one year with fault-tolerant software like that proposed by Vega.

Fomalhaut makes some use of modern design practices by experienced personnel. They have extensive tools to support Arcturus' design, including on APSE, but only have basic maxi tools to support Vega's design. They have about three years' experience with DOD-STD-2167A development practices.

Fomalhaut will be performing support at a single location and with a single organization, although personnel may be only 50% dedicated to the C-28 C&D CSCI effort. They view the project as a mixture of new and repeated effort. Integration is viewed as a closely-coupled effort with a mix of personnel assigned. Fomalhaut's office environment is not always conducive to productivity, since it is an open bay area with little privacy in a renovated civil war prison.

Also, documentation capability is weak - only partial automation of graphics and text support exists, and the programmers have little or no automated documentation capability. Despite the limitations, AFLC has found Fomalhaut to be extremely conscientious and has a good track record.

Fomalhaut's labor rate is relatively low. The average salary is $59 per hour, or $9,000 per month. All other information about the project is the same as for the developer, Arcturus or realize the 10% savings from use of fault-tolerance.

## Appendix E: COCOMO Formulas and Effort Multipliers

(Boehm, 1981:352-355)

### COCOMO Formulas

The Detailed COCOMO model used to develop the expert system in this study. The formulas are as follows:

Software Mode

Organic $(MM)_{NOM} = 3.2(KDSI)^{1.05}$ (1)

$TDEV = 2.5(MM)_{DEV}^{0.38}$ (2)

Semi-Detached $(MM)_{NOM} = 3.0(KDSI)^{1.12}$ (3)

$TDEV = 2.5(MM)_{DEV}^{0.35}$ (4)

Embedded $(MM)_{NOM} = 2.8(KDSI)^{1.20}$ (5)

$TDEV = 2.5(MM)_{DEV}^{0.32}$ (6)

where,

$(MM)_{NOM}$ is man-months nominal

KDSI is thousand of delivered source instructions

TDEV is development schedule

$(MM)_{DEV}$ is development time in man-months

Adaptation Adjustment Factor (AAF)

AAF = 0.4 (%t design modified) + .3 (% code modified)

+ .3 (% integration modified) (7)

Equivalent Delivered Source Instructions (EDSI)

EDSI = (Adapted DSI) (AAF / 100) (8)

where,

Adapted DSI is adapted delivered source instructions

AAF is adaptation adjustment factor

## Nominal Phase Distribution of Effort

$$(MM)_{NOM_p} = [(EDSI)\,(FRAC)_p\,] \div [(EDSI \div (MM)_{NOM})] \quad (9)$$

where,

$(MM)_{NOM_p}$ nominal phase effort the module in man months

EDSI is equivalent delivered source instructions

$(FRAC)_p$ is phase distribution of effort

$(MM)_{NOM}$ is nominal effort in man months

## Effort Multipliers

The following abbreviations are used in the subsequent listing to describe the different phases of software development. Also, the specific product, computer, personnel, and project cost drivers will be listed by attribute.

### Program Phases

RPD - Requirements and Product Design
DD - Detailed Design
CUT - Code and Unit Test
IT - Integration and Test

### Detailed COCOMO Cost Drivers

#### Product Attributes

RELY - Required software reliability
DATA - Data base size
CPLX - Software product complexity

#### Computer Attributes

TIME - Execution time constraint
STOR - Main storage constraint
VIRT - Virtual machine volatility
TURN - Computer turnaround time

## Personnel Attributes

ACAP - Analyst capability
AEXP - Applications experience
PCAP - Programmer capability
VEXP - Virtual machine experience
LEXP - Language experience

## Project Attributes

MODP - Use of modern programming practices
TOOL - Use of software tools
SCED - Development schedule constraint

## COCOMO Effort Multipliers

| ACAP | RPD | DD | CUT | IT |
|---|---|---|---|---|
| VERY LOW | 1.80 | 1.35 | 1.35 | 1.50 |
| LOW | 1.35 | 1.15 | 1.15 | 1.20 |
| NOMINAL | 1.00 | 1.00 | 1.00 | 1.00 |
| HIGH | 0.75 | 0.90 | 0.90 | 0.85 |
| VERY HIGH | 0.55 | 0.75 | 0.75 | 0.70 |

| AEXP | RPD | DD | CUT | IT |
|---|---|---|---|---|
| VERY LOW | 1.40 | 1.30 | 1.25 | 1.25 |
| LOW | 1.20 | 1.15 | 1.10 | 1.10 |
| NOMINAL | 1.00 | 1.00 | 1.00 | 1.00 |
| HIGH | 0.87 | 0.9 | 0.92 | 0.92 |
| VERY HIGH | 0.75 | 0.80 | 0.85 | 0.85 |

| CPLX | RPD | DD | CUT | IT |
|---|---|---|---|---|
| VERY LOW | 0.70 | 0.70 | 0.70 | 0.70 |
| LOW | 0.85 | 0.85 | 0.85 | 0.85 |
| NOMINAL | 1.00 | 1.00 | 1.00 | 1.00 |
| HIGH | 1.15 | 1.15 | 1.15 | 1.15 |
| VERY HIGH | 1.30 | 1.30 | 1.30 | 1.30 |
| EXTRA HIGH | 1.65 | 1.65 | 1.65 | 1.65 |

| DATA | RPD | DD | CUT | IT |
|---|---|---|---|---|
| LOW | 0.95 | 0.95 | 0.95 | 0.90 |
| NOMINAL | 1.00 | 1.00 | 1.00 | 1.00 |
| HIGH | 1.10 | 1.05 | 1.05 | 1.15 |
| VERY HIGH | 1.20 | 1.10 | 1.10 | 1.30 |

| LEXP | RPD | DD | CUT | IT |
|---|---|---|---|---|
| VERY LOW | 1.02 | 1.10 | 1.20 | 1.20 |
| LOW | 1.00 | 1.05 | 1.10 | 1.10 |
| NOMINAL | 1.00 | 1.00 | 1.00 | 1.00 |
| HIGH | 0.90 | 0.98 | 0.92 | 0.92 |

| MODP | RPD | DD | CUT | IT |
|---|---|---|---|---|
| VERY LOW | 1.05 | 1.10 | 1.25 | 1.50 |
| LOW | 1.00 | 1.05 | 1.10 | 1.20 |
| NOMINAL | 1.00 | 1.00 | 1.00 | 1.00 |
| HIGH | 1.00 | 0.95 | 0.90 | 0.83 |
| VERY HIGH | 1.00 | 0.90 | 0.80 | 0.65 |

| PCAP | RPD | DD | CUT | IT |
|---|---|---|---|---|
| VERY LOW | 1.00 | 1.50 | 1.50 | 1.50 |
| LOW | 1.00 | 1.20 | 1.20 | 1.20 |
| NOMINAL | 1.00 | 1.00 | 1.00 | 1.00 |
| HIGH | 1.00 | 0.83 | 0.83 | 0.83 |
| VERY HIGH | 1.00 | 0.65 | 0.65 | 0.65 |

| RELY | RPD | DD | CUT | IT |
|---|---|---|---|---|
| VERY LOW | 0.80 | 0.80 | 0.80 | 0.60 |
| LOW | 0.90 | 0.90 | 0.90 | 0.80 |
| NOMINAL | 1.00 | 1.00 | 1.00 | 1.00 |
| HIGH | 1.10 | 1.10 | 1.10 | 1.30 |
| VERY HIGH | 1.30 | 1.30 | 1.30 | 1.70 |

| SCED | RPD | DD | CUT | IT |
|---|---|---|---|---|
| VERY LOW | 1.10 | 1.25 | 1.25 | 1.25 |
| LOW | 1.00 | 1.15 | 1.15 | 1.10 |
| NOMINAL | 1.00 | 1.00 | 1.00 | 1.00 |
| HIGH | 1.10 | 1.10 | 1.00 | 1.00 |
| VERY HIGH | 1.15 | 1.15 | 1.05 | 1.05 |

| STOR | RPD | DD | CUT | IT |
|---|---|---|---|---|
| NOMINAL | 1.00 | 1.00 | 1.00 | 1.00 |
| HIGH | 1.05 | 1.05 | 1.05 | 1.10 |
| VERY HIGH | 1.2 | 1.15 | 1.15 | 1.35 |
| EXTRA HIGH | 1.55 | 1.45 | 1.45 | 1.85 |

| TIME | RPD | DD | CUT | IT |
|---|---|---|---|---|
| NOMINAL | 1.00 | 1.00 | 1.00 | 1.00 |
| HIGH | 1.10 | 1.10 | 1.10 | 1.15 |
| VERY HIGH | 1.30 | 1.25 | 1.25 | 1.40 |
| EXTRA HIGH | 1.65 | 1.55 | 1.55 | 1.95 |

| TOOL | RPD | DD | CUT | IT |
|---|---|---|---|---|
| VERY LOW | 1.02 | 1.05 | 1.35 | 1.45 |
| LOW | 1.00 | 1.02 | 1.15 | 1.20 |
| NOMINAL | 1.00 | 1.00 | 1.00 | 1.00 |
| HIGH | 0.98 | 0.95 | 0.90 | 0.85 |
| VERY HIGH | 0.95 | 0.90 | 0.80 | 0.70 |

| TURN | RPD | DD | CUT | IT |
|---|---|---|---|---|
| LOW | 0.98 | 0.95 | 0.70 | 0.90 |
| NOMINAL | 1.00 | 1.00 | 1.00 | 1.00 |
| HIGH | 1.00 | 1.00 | 1.10 | 1.15 |
| VERY HIGH | 1.02 | 1.05 | 1.20 | 1.30 |

| VEXP | RPD | DD | CUT | IT |
|---|---|---|---|---|
| VERY LOW | 1.10 | 1.10 | 1.30 | 1.30 |
| LOW | 1.05 | 1.05 | 1.15 | 1.15 |
| NOMINAL | 1.00 | 1.00 | 1.00 | 1.00 |
| HIGH | 0.90 | 0.90 | 0.90 | 0.90 |

| VIRT | RPD | DD | CUT | IT |
|---|---|---|---|---|
| LOW | 0.95 | 0.90 | 0.85 | 0.80 |
| NOMINAL | 1.00 | 1.00 | 1.00 | 1.00 |
| HIGH | 1.10 | 1.12 | 1.15 | 1.20 |
| VERY HIGH | 1.20 | 1.25 | 1.30 | 1.40 |

# Appendix F: Expert Application Questionnaire Responses

This questionnaire has been written to obtain expert knowledge on software cost estimation. In particular, the questions will conform to the parameters identified in the Detailed COCOMO software cost estimation model with additional queries for areas not covered by COCOMO. Those areas not covered by COCOMO will be highlighted with boldface type. In almost all cases the answering of the question will be followed by a "why" question, in order to clarify any misunderstandings that may occur.

Interviewee: Professor Dan Ferens, AFIT/LSY.
Years of applied cost estimation: Seven.
Years teaching cost estimation: Six.

Q1: How are you presently estimating software development costs?

Ans: I presently use REVIC extensively for classroom exercises and to obtain rough order of magnitude estimates of cost. This model, however, is not the most extensive and does have limitations.

Q2: Do you feel there are some models that do a better job than others?

Ans: SEER has been proven very effective at ESD (Electronic Systems Division) in evaluating ground based avionic systems. Price-S, in a similar fashion, has been proven effective for Aeronautical Systems Division (ASD) airborne systems. For schedule computations, SEER and Price-S provide fairly consistent estimates. SEER appears to have a greater number of input, and an ESD study showed SEER to be accurate to within 10% for ground-based projects.

Q3: What do you consider as the core requirements for estimating software cost? (ie, SLOC, function points, etc).

Ans: Source Lines of Code is the primary determinant used in almost all the models that I use. Function points are questionable because people don't know what they are or how to use them. *Are there any others that you consider important to use?* Ans: Other requirements that could be used if validated would be:

- Specification Integration of HW and SW.
- Environmental security.

- Language use such as ADA or other HOLs.

I don't like to consider personnel experience because it is difficult to quantify and verify, although it is a cost driver.

Q4: Are there any algorithms or heuristics that you use or recommend, for ascertaining development cost?

Ans: These are really situationally dependent. Some models lend themselves better to some applications and others to others.

Q5: Are there any non-cost variables to consider when evaluating a software proposal?

Ans: Feasibility of technical design or risk, management structure (such as can occur in joint ventures), and people stability and quality.

Q6 What is the impact of the above considerations on cost, if any?

Ans: Again it depends on the situation.

Q7: How do you decide which Mode the software should be characterized as?

Ans: Use the COCOMO matrix as identified by Boehm in his book on software cost estimation. Just as importantly, however, talk to the contractor and ensure what he said in the proposal is actually the level of system complexity and design he is building to.

COST DRIVERS

*An overall comment about cost drivers. This interviewee felt that the quantification of these variables was best defined by the guidance provided in the model handbook. Each model provides a qualitative or quantitative continuum for each attribute with a corollary numeric value for computation. The expert evaluation comes in when analyzing the proposal and selecting the qualitative level of application.*

Q8: What variables do you consider when evaluating a proposal for Personnel attributes.

Ans: In general for all cost drivers, I look for as fine a detail for each attribute as possible. Some models, such as SEER, provide that detail and others do not. If you can break your environment down into greater detail, such as a continuum from organic through embedded and even including HOL's such as ADA, your output will be closer to what you will see in the field. Finer detail in management variability, past history, and impacts of factors like geographical separation will aid in decreasing the variability of your output.

Q9: What indicators do you use to rate the Analysts capabilities (ACAP)?

Ans: This is rated as a percentile which is derived from the model's, in this case COCOMO, description for the attribute.

Q10: What indicators do you use to rate the development team's Application experience (AEXP)?

Ans: Use the model description for this variable.

Q11: What indicators do you use to rate the Programmer capabilities (PCAP)?

Ans: Use the proposal and past history to get a feeling for their potential performance. Interview the contractor to find out who will make up the team and how long they are expected to stay with the effort. Team integrity and overall time working with the team is also important. These concepts apply to all team oriented attributes. This also applies to ACAP, AEXP, VEXP, and LEXP.

Q12: What indicators do you use to rate the development team's Virtual machine experience (VEXP)?

Ans: Use the model description for this variable.

Q13: What indicators do you use to rate the teams's experience with the programming language (LEXP)?

Ans: Use the model description for this variable.

Q14: Are there any other personnel attributes you consider as important in determining software cost?

Ans: Perhaps turnover, it can be predicted.

Q15: What variables do you consider when evaluating a proposal for Project attributes?

Ans: Those identified as salient by the model for this attribute.

Q16: How do you rate the use of modern programming practices (MODP)?

Ans: In most cases I rate the use of MODP as HI. This is because most contractors have gone to using practices as defined in DoD Std-2167A. This category also lends itself to finer discrimination if that is required.

Q17: What ratings do you use when evaluating the proposal for the use of software tools (TOOL)?

Ans: The same logic applies to TOOLS as indicated in the last question.

Q18: How do you rate the impact of schedule constraints on software development cost (SCED)?

Ans: I let the model make the determination for me. Price-S, for example, is a schedule oriented estimation package. It allows for schedule generation and provides the user the capability of doing "what if" evaluations of the proposal. Conversely, REVIC is limited in what it can do but does provide fairly consistent results.

Q19: How is managerial risk rated (**RISK**)?

Ans: I perceive this to be a management reserve computation which is normally a straightforward multiplier applied against the bottom line results of the computations. Risk parameters have been poorly defined in the past and really need more historical data for support before this variable can be use effectively.

Q20: How do you rate the impact of classified development (**SECU**)?

Ans: Presently there are only two levels in REVIC, either classified or unclassified. This makes the evaluation fairly easy but it doesn't take into effect the different levels of classified that have the potential for driving cost even higher. This category needs finer detail.

Q21: What variables do you consider when evaluating requirements volatility (**RVOL**)?

Ans: Requirements volatility reflects the stability of the user's requirements. I would evaluate this category by reviewing the history of this type of development and also the stability of the user in staying with the original requirements.

Q22: Are there any other project attributes you consider important in determining software cost?

Ans: Just those dictated by the model in use.

Q23: What variables do you consider when evaluating a proposal for product attributes?

Ans: Those required by the model.

Q24: How do you rate the impact of software reliability (RELY)?

Ans: I use the continuum as defined in the model. SEER breaks this category down into finer detail which makes it more effective.

Q25: What do you consider when rating the data base size (DATA)?

Ans: I skip this one in most cases. Not many people know what it means plus it doesn't effect the evaluation that much. Just use a nominal rating when evaluating weapons systems.

Q26: How do you select the values for product complexity (CPLX)?

Ans: Product complexity is best evaluated by using the model complexity algorithms

Q27: What variables are considered when rating required reusability **(RUSE)**?

Ans: This is another questionable category. Not many developers know if the program is going to be reused or not. This rating could also fall under complexity.

Q28: Are there any other product attributes you consider important in determining software cost?

Ans: Not at this time.

Q29 What variables do you consider when evaluating a proposal for Computer attributes?

Ans: Again, those which the model considers important for evaluation.

Q30: What criteria do you use when rating computer time constraints (TIME)?

Ans: The values identified by the model in use.

Q31 How do you measure the impact of system storage (STOR) limitations on development cost?

Ans: The values identified by the model in use.

Q32: What measures do you use when rating the virtual machine volatility (VIRT)?

Ans: The values identified by the model in use.

Q33: How do you rate the development center's ability to turnaround output (TURN)?

Ans: The values identified by the model in use.

Q34: Are there any other computer attributes you consider important in determining software cost?

Ans: I consider the level of interactive development as an important computer attribute. If it isn't interactive, given today's technology, I would want the contractor to tell me why.

OTHER COST DRIVERS

Q35: How do you compute the development schedule for each effort?

Ans: I don't use any method other than that provided by the cost estimation package.

Q36: What considerations do you make when calibrating your model?

Ans: In order to effectively calibrate the model you need to have the historical data for the area you wish to calibrate. Contractor performance, environment, and application are a few areas that can be calibrated. Contractor performance is difficult do the standardized procedure of sanitizing performance reports.

Q37: How do you rate the proposal for software support or maintenance costs?

Ans: Maintenance costs are usually difficult to realistically evaluate during the development period. Most models provide maintenance costs as a part of their design, but have only been effective in estimating relative costs.

Q38: Does the use of ADA have an impact on your determination of cost?

Ans: In REVIC, as an example, ADA development has its own algorithms. New languages, such as NOMAD which is a very high order language (VHOL), would have to have their own multipliers or algorithms.

Q39: Are there any special cases, such as the development of an expert system, that would change any of the above variables?

Ans: New languages and environments could have the potential for changing any of the above variables, categories or attributes. They also have the potential for creating new ones to make up for their unique requirements.

Q40: How do you determine total life-cycle cost?

Ans: Total LCC is computed by the models. They are the sum of the development and support costs.

## Interview II

Interviewee: Captain Gregory Pshsnychyniak, AFIT/LSY.
Years of applied cost estimation: Three in airborne avionics.

Q1: How are you presently estimating software development costs?
Ans: We are presently using Price-S and REVIC in our cost estimations. Sometime in the near future we will probably begin using SEER.

Q2: Do you feel there are some models that do a better job than others?
Ans: I feel SEER will do a better job. Price-S does a good job for us now considering the program phase is pre-FSD. Price-S tends to generalize factors like personnel attributes and tools so you don't get caught up in unnecessary detail early on in the program.

Q3: What do you consider as the core requirements for estimating software cost? (ie, SLOC, function points, etc).
Ans: Source lines of code and understanding of the software application is vital. Complexity of the task and integration with hardware is also essential.

Q4: Are there any algorithms or heuristics that you use or recommend, for ascertaining development cost?
Ans: The only algorithms we use other than those in the models themselves is the Wolverton equation for determining man month estimates.

Q5: Are there any non-cost variables to consider when evaluating a software proposal?
Ans: Risk is probably one of the more subjective variables to evaluate. Presently though, it is difficult to define and evaluate.

Q6 What is the impact of the above considerations on cost, if any?
Ans: All of the variables have an impact. Defense of my software estimates will come from the engineering understanding of the terms in question.

Q7: How do you decide which Mode the software should be characterized as?
Ans: When using Price-S I go to the subsystem level tie the mode to phase of development. Price-S is broken down by CSCI, CSC, etc. Revic on

phase of development. Price-S is broken down by CSCI, CSC, etc. Revic on the other hand uses the classical modes of organic, semi-detached, and embedded. When using REVIC I always select embedded due to the complexity of our programs.

COST DRIVERS

*An overall comment about cost drivers. This interviewee felt that the quantification of these variables was best defined by the guidance provided in the model handbook. When questions arose that were not covered in the manual, then the local technical representative from General Electric, for Price-S, would be queried to ascertain the appropriate value for the problem at hand. Further, he felt that some drivers, such as data-base size, had no effect on the outcome of the computation and would set those values at nominal.*

Q8: What variables do you consider when evaluating a proposal for Personnel attributes.

Ans: The overall experience of the contractor and the development team in the use of ADA is important. That experience has a direct effect on ACAP, AEXP, LEXP, PCAP, etc.

Q9: What indicators do you use to rate the Analysts capabilities (ACAP)?

Ans: Basically, because we use Price-S rather extensively, we use what the model recommends for all the personnel attributes.

Q10: What indicators do you use to rate the development team's Application experience (AEXP)?

Ans: Use the model description for this variable.

Q11: What indicators do you use to rate the Programmer capabilities (PCAP)?

Ans: Use the model description for this variable.

Q12: What indicators do you use to rate the development team's Virtual machine experience (VEXP)?

Ans: We take a hard look at the time the virtual machine has been in place and the experience of the developing team in using it.

Q13: What indicators do you use to rate the teams's experience with the programming language (LEXP)?
Ans: Use the model description for this variable.

Q14: Are there any other personnel attributes you consider as important in determining software cost?
Ans: None at this time.

Q15: What variables do you consider when evaluating a proposal for Project attributes?
Ans: We use what the model recommends explicitly. If not you end up with an "apple and oranges" type of comparison and thereby lose consistency of your results.

Q16: How do you rate the use of modern programming practices (MODP)?
Ans: If the contractor is complying with 2167 we will rate them fairly high. We also evaluate how long they have been using these practices and the experience of the development team in using them. This probably goes back to personnel attributes again.

Q17: What ratings do you use when evaluating the proposal for the use of software tools (TOOL)?
Ans: Its hard to separate tools from MODP. We rate them in a similar fashion.

Q18: How do you rate the impact of schedule constraints on software development cost (SCED)?
Ans: I let the model make the determination for me. However, for better results it is necessary to provide a baseline or beginning point for the program to begin.

Q19: How is managerial risk rated (**RISK**)?
Ans: We evaluate the contractor under one very simple criteria, do we believe they can do the job. This evaluation is based in part on the intensity of the effort.

Q20: How do you rate the impact of classified development (**SECU**)?
Ans: Use the model description for this variable.

Q21: What variables do you consider when evaluating requirements volatility (**RVOL**)?

Ans: Requirements volatility goes above and beyond what the model calls for. Freezing the requirements to ensure you have a baseline to design to is essential for good estimation. We in the SPO control that to a certain extent and prompt our users to that effect as necessary.

Q22: Are there any other project attributes you consider important in determining software cost?

Ans: Use the model description for this variable.

Q23: What variables do you consider when evaluating a proposal for product attributes?

Ans: Use the model description for this variable.

Q24: How do you rate the impact of software reliability (RELY)?

Ans: Use the model description for this variable.

Q25: What do you consider when rating the data base size (DATA)?

Ans: This one doesn't effect us at all. We set it to nominal.

Q26: How do you select the values for product complexity (CPLX)?

Ans: Complexity is one variable we are very concerned about. In our system the values are invariably high due to the integration required of different software modules and their potential effect on system operation.

Q27: What do you consider when rating required reusability (**RUSE**)?

Ans: We are looking at the reusability inherent in designing a program using ADA. Airframe, system, subsystem, component modules , are all subject to being reused. We are not looking at reuse of our software on other programs, though, just within our system, the ATF.

Q28: Are there any other product attributes you consider important in determining software cost?

Ans: Not at this time.

Q29 What variables do you consider when evaluating a proposal for Computer attributes?

Ans: Again, those which the model considers important for evaluation.

Q30: What criteria do you use when rating computer time constraints (TIME)?

Ans: Because we specified 50% below specified values this variable will always be rated low.

Q31 How do you measure the impact of system storage (STOR) limitations on development cost?

Ans: Same as for TIME.

Q32: What measures do you use when rating the virtual machine volatility (VIRT)?

Ans: The values identified by the model in use.

Q33: How do you rate the development center's ability to turnaround output (TURN)?

Ans: The values identified by the model in use.

Q34: Are there any other computer attributes you consider important in determining software cost?

Ans: None at this time.

OTHER COST DRIVERS

Q35: How do you compute the development schedule for each effort?

Ans: We don't do the schedule other than that provided in the model. Our input is provided to the managers of each particular program element and they make up the schedules.

Q36: What considerations do you make when calibrating your model?

Ans: We don't calibrate because it isn't necessary. We leave the models alone to ensure that each contractor is measured using the same variables and algorithms. Otherwise it could potentially bias the results of our evaluation.

Q37: How do you rate the proposal for software support or maintenance costs?

Ans: Use the model description for this variable.

Q38: Does the use of ADA have an impact on your determination of cost?

Ans: Absolutely. As mentioned earlier ADA has an impact on size, speed, development time, and a variety of other variables.

Q39: Are there any special cases, such as the development of an expert system, that would change any of the above variables?

Ans: None that I am familiar with.

Q40: How do you determine total life-cycle cost?

Ans: Even though the models can provide total LCC, I don't concern myself with it at this time because I look at development cost only. Logistics and engineering look at the total life-cycle cost.

## Appendix G: Expert System Rule Listing
## Nexpert Object™

```
RULE : Rule 1
If
    there is evidence of Project.Start
Then Action1
    is confirmed.
    And 0.0 is assigned to n
    And 0.0 is assigned to p
    And 0.0 is assigned to subedsi

RULE : Rule 2
If
    there is no evidence of |SUBSYSTEM|.Requirement
Then Action10
    is confirmed.
    And subedsi is assigned to <|SUBSYSTEM|>.EDSI
    And |MODE|.Conform_ext_int_spec is assigned to ceis
    And |MODE|.Lines_of_Code is assigned to loc
    And |MODE|.Need_innov_data_proc is assigned to nidp
    And |MODE|.Product_complexity is assigned to pc
    And |MODE|.Schedule_adherence is assigned to sa
    And |MODE|.Concurrent_dev_of_new_hw_and_oper_proced is assigned to cdnh
    And |MODE|.Conform_preestablish_req is assigned to cpr
    And (ceis)+(loc)+(nidp)+(pc)+(sa)+(cdnh)+(cpr) is assigned to |MODE|.Worth

RULE : Rule 3
If
    |SUBSYSTEM|.Sced is "Stretched_out_development_schedule"
Then Action100
    is confirmed.
    And 1.1 is assigned to <|SUBSYSTEM|>.SCEDpd
    And 1.1 is assigned to <|SUBSYSTEM|>.SCEDdd
    And 1.0 is assigned to <|SUBSYSTEM|>.SCEDcut
    And 1.0 is assigned to <|SUBSYSTEM|>.SCEDit
    And <|SUBSYSTEM|>.Sced_rating is set to "High"

RULE : Rule 4
If
    |SUBSYSTEM|.Sced is "Very_stretched_out_development_schedule"
Then Action101
    is confirmed.
    And 1.15 is assigned to <|SUBSYSTEM|>.SCEDpd
    And 1.15 is assigned to <|SUBSYSTEM|>.SCEDdd
    And 1.05 is assigned to <|SUBSYSTEM|>.SCEDcut
    And 1.05 is assigned to <|SUBSYSTEM|>.SCEDit
    And <|SUBSYSTEM|>.Sced_rating is set to "Very_High"
```

RULE : Rule 5
If
    edsi is greater than 0.0
Then Action102
    is confirmed.
    And
<|SUBSYSTEM|>.RELYpd*<|SUBSYSTEM|>.STORpd*<|SUBSYSTEM|>.VIRTpd*<|SUBSYSTEM|>.TURNpd*<|SUBSYSTEM|>.ACAPpd*<|SUBSYSTEM|>.MODPpd*<|SUBSYSTEM|>.TOOLpd*<|SUBSYSTEM|>.SCEDpd is assigned to <|SUBSYSTEM|>.EAFsspd
    And
<|SUBSYSTEM|>.RELYdd*<|SUBSYSTEM|>.STORdd*<|SUBSYSTEM|>.VIRTdd*<|SUBSYSTEM|>.TURNdd*<|SUBSYSTEM|>.ACAPdd*<|SUBSYSTEM|>.MODPdd*<|SUBSYSTEM|>.TOOLdd*<|SUBSYSTEM|>.SCEDdd is assigned to <|SUBSYSTEM|>.EAFssdd
    And
<|SUBSYSTEM|>.RELYcut*<|SUBSYSTEM|>.STORcut*<|SUBSYSTEM|>.VIRTcut*<|SUBSYSTEM|>.TURNcut*<|SUBSYSTEM|>.ACAPcut*<|SUBSYSTEM|>.MODPcut*<|SUBSYSTEM|>.TOOLcut*<|SUBSYSTEM|>.SCEDcut is assigned to <|SUBSYSTEM|>.EAFsscut
    And
<|SUBSYSTEM|>.RELYit*<|SUBSYSTEM|>.STORit*<|SUBSYSTEM|>.VIRTit*<|SUBSYSTEM|>.TURNit*<|SUBSYSTEM|>.ACAPit*<|SUBSYSTEM|>.MODPit*<|SUBSYSTEM|>.TOOLit*<|SUBSYSTEM|>.SCEDit is assigned to <|SUBSYSTEM|>.EAFssit
    And <|SUBSYSTEM|>.MMmodpd*<|SUBSYSTEM|>.EAFsspd is assigned to <|SUBSYSTEM|>.MMestpd
    And <|SUBSYSTEM|>.MMmoddd*<|SUBSYSTEM|>.EAFssdd is assigned to <|SUBSYSTEM|>.MMestdd
    And <|SUBSYSTEM|>.MMmodcut*<|SUBSYSTEM|>.EAFsscut is assigned to <|SUBSYSTEM|>.MMestcut
    And <|SUBSYSTEM|>.MMmodit*<|SUBSYSTEM|>.EAFssit is assigned to <|SUBSYSTEM|>.MMestit
    And
<|SUBSYSTEM|>.MMestpd+<|SUBSYSTEM|>.MMestdd+<|SUBSYSTEM|>.MMestcut+<|SUBSYSTEM|>.MMestit is assigned to <|SUBSYSTEM|>.MMesttot

RULE : Rule 6
If
    <|SUBSYSTEM|>.Modename is "ORGANIC"
Then Action103
    is confirmed.
    And (2.5*<|SUBSYSTEM|>.MMesttot).(0.38) is assigned to <|SUBSYSTEM|>.TDEV

RULE : Rule 7
If
    <|SUBSYSTEM|>.Modename is "SEMIDETACHED"
Then Action104
    is confirmed.
    And (2.5*<|SUBSYSTEM|>.MMesttot).(0.35) is assigned to <|SUBSYSTEM|>.TDEV

RULE : Rule 8
If
    <|SUBSYSTEM|>.Modename is "EMBEDDED"
Then Action105

```
RULE : Rule 8 (cont.)
    is confirmed.
    And (2.5*‹|SUBSYSTEM|›.MMesttot),(0.32) is assigned to ‹|SUBSYSTEM|›.TDEV

RULE : Rule 9
If
    edsi is greater than 0.0
Then Action106
    is confirmed.
    And |SUBSYSTEM|.AA__44__MM_kpd is assigned to AA__4__mm_kpd
    And |SUBSYSTEM|.AA__42__MM_kdd is assigned to AA__2__mm_kdd
    And |SUBSYSTEM|.AA__41__MM_kcut is assigned to AA__1__mm_kcut
    And |SUBSYSTEM|.AA__43__MM_kit is assigned to AA__3__mm_kit
    And AA__4__mm_kpd is assigned to ‹|SUBSYSTEM|›.AA__44__MM_kpd
    And AA__2__mm_kdd is assigned to ‹|SUBSYSTEM|›.AA__42__MM_kdd
    And AA__1__mm_kcut is assigned to ‹|SUBSYSTEM|›.AA__41__MM_kcut
    And AA__3__mm_kit is assigned to ‹|SUBSYSTEM|›.AA__43__MM_kit
    And AA__4__mm_kpd*‹|SUBSYSTEM|›.MMestpd is assigned to
‹|SUBSYSTEM|›.AA__8___Kpd
    And AA__2__mm_kdd*‹|SUBSYSTEM|›.MMestdd is assigned to
‹|SUBSYSTEM|›.AA__6___Kdd
    And AA__1__mm_kcut*‹|SUBSYSTEM|›.MMestcut is assigned to
‹|SUBSYSTEM|›.AA__5___Kcut
    And AA__3__mm_kit*‹|SUBSYSTEM|›.MMestit is assigned to
‹|SUBSYSTEM|›.AA__7___Kit
    And
‹|SUBSYSTEM|›.AA__8___Kpd+‹|SUBSYSTEM|›.AA__6___Kdd+‹|SUBSYSTEM|›.AA__5___Kcut
+‹|SUBSYSTEM|›.AA__7___Kit is assigned to ‹|SUBSYSTEM|›.AA__10___Ktot
    And ‹|SUBSYSTEM|›.EDSI/‹|SUBSYSTEM|›.MMesttot is assigned to
‹|SUBSYSTEM|›.EDSIpMM
    And ‹|SUBSYSTEM|›.AA__10___Ktot/‹|SUBSYSTEM|›.EDSI is assigned to
‹|SUBSYSTEM|›.AA__9___KpEDSI

RULE : Rule 10
If
    |MODE|.Worth is greater than or equal to 0.0
    And |MODE|.Worth is less than 20.0
Then Action11
    is confirmed.
    And (3.2)*(‹|SUBSYSTEM|›.EDSI/1000.0),(1.05) is assigned to MMnom
    And |MODE|.Name is set to "ORGANIC"
    And ‹|SUBSYSTEM|›.Modename is set to "ORGANIC"
    And ‹|SUBSYSTEM|›.EDSI/MMnom is assigned to EDSIMMnom

RULE : Rule 11
If
    |MODE|.Worth is greater than or equal to 20.0
    And |MODE|.Worth is less than 45.0
Then Action12
    is confirmed.
    And (3.0)*(‹|SUBSYSTEM|›.EDSI/1000.0),(1.12) is assigned to MMnom
```

RULE : Rule 11 (cont.)
And |MODE|.Name is set to "SEMIDETACHED"
And <|SUBSYSTEM|>.Modename is set to "SEMIDETACHED"
And <|SUBSYSTEM|>.EDSI/MMnom is assigned to EDSIMMnom

RULE : Rule 12
If
|MODE|.Worth is greater than or equal to 45.0
And |MODE|.Worth is less than or equal to 70.0
Then Action13
is confirmed.
And (2.8)*(<|SUBSYSTEM|>.EDSI/1000.0),(1.2) is assigned to MMnom
And |MODE|.Name is set to "EMBEDDED"
And <|SUBSYSTEM|>.Modename is set to "EMBEDDED"
And <|SUBSYSTEM|>.EDSI/MMnom is assigned to EDSIMMnom

RULE : Rule 13
If
<|SUBSYSTEM|>.EDSI is less than or equal to 2000.0
And <|SUBSYSTEM|>.Modename is "ORGANIC"
Then Action20
is confirmed.
And 0.16 is assigned to FRACpd
And 0.26 is assigned to FRACdd
And 0.42 is assigned to FRACcut
And 0.16 is assigned to FRACit
And (<|SUBSYSTEM|>.EDSI*FRACpd)/EDSIMMnom is assigned to MMnompd
And (<|SUBSYSTEM|>.EDSI*FRACdd)/EDSIMMnom is assigned to MMnomdd
And (<|SUBSYSTEM|>.EDSI*FRACcut)/EDSIMMnom is assigned to MMnomcut
And (<|SUBSYSTEM|>.EDSI*FRACit)/EDSIMMnom is assigned to MMnomit

RULE : Rule 14
If
edsi is greater than 2000.0
And edsi is less than or equal to 8000.0
And |MODE|.Name is "ORGANIC"
Then Action21
is confirmed.
And 0.16 is assigned to FRACpd
And 0.25 is assigned to FRACdd
And 0.4 is assigned to FRACcut
And 0.19 is assigned to FRACit
And (edsi*FRACpd)/EDSIMMnom is assigned to MMnompd
And (edsi*FRACdd)/EDSIMMnom is assigned to MMnomdd
And (edsi*FRACcut)/EDSIMMnom is assigned to MMnomcut
And (edsi*FRACit)/EDSIMMnom is assigned to MMnomit

RULE : Rule 15
If
edsi is greater than 8000.0
And edsi is less than or equal to 32000.0

RULE : Rule 15 (cont.)
And |MODE|.Name is "ORGANIC"
Then Action22
    is confirmed.
    And 0.16 is assigned to FRACpd
    And 0.24 is assigned to FRACdd
    And 0.38 is assigned to FRACcut
    And 0.22 is assigned to FRACit
    And (edsi*FRACpd)/EDSIMMnom is assigned to MMnompd
    And (edsi*FRACdd)/EDSIMMnom is assigned to MMnomdd
    And (edsi*FRACcut)/EDSIMMnom is assigned to MMnomcut
    And (edsi*FRACit)/EDSIMMnom is assigned to MMnomit

RULE : Rule 16
If
    edsi is greater than 32000.0
    And edsi is less than or equal to 128000.0
    And |MODE|.Name is "ORGANIC"
Then Action23
    is confirmed.
    And 0.16 is assigned to FRACpd
    And 0.23 is assigned to FRACdd
    And 0.36 is assigned to FRACcut
    And 0.25 is assigned to FRACit
    And (edsi*FRACpd)/EDSIMMnom is assigned to MMnompd
    And (edsi*FRACdd)/EDSIMMnom is assigned to MMnomdd
    And (edsi*FRACcut)/EDSIMMnom is assigned to MMnomcut
    And (edsi*FRACit)/EDSIMMnom is assigned to MMnomit

RULE : Rule 17
If
    edsi is less than or equal to 2000.0
    And |MODE|.Name is "SEMIDETACHED"
Then Action24
    is confirmed.
    And 0.17 is assigned to FRACpd
    And 0.27 is assigned to FRACdd
    And 0.37 is assigned to FRACcut
    And 0.19 is assigned to FRACit
    And (edsi*FRACpd)/EDSIMMnom is assigned to MMnompd
    And (edsi*FRACdd)/EDSIMMnom is assigned to MMnomdd
    And (edsi*FRACcut)/EDSIMMnom is assigned to MMnomcut
    And (edsi*FRACit)/EDSIMMnom is assigned to MMnomit

RULE : Rule 18
If
    edsi is greater than 2000.0
    And edsi is less than or equal to 8000.0
    And |MODE|.Name is "SEMIDETACHED"
Then Action25
    is confirmed.

RULE : Rule 18 (cont.)
    And 0.17 is assigned to FRACpd
    And 0.26 is assigned to FRACdd
    And 0.35 is assigned to FRACcut
    And 0.22 is assigned to FRACit
    And (edsi*FRACpd)/EDSIMMnom is assigned to MMnompd
    And (edsi*FRACdd)/EDSIMMnom is assigned to MMnomdd
    And (edsi*FRACcut)/EDSIMMnom is assigned to MMnomcut
    And (edsi*FRACit)/EDSIMMnom is assigned to MMnomit

RULE : Rule 19
If
    edsi is greater than 8000.0
    And edsi is less than or equal to 32000.0
    And |MODE|.Name is "SEMIDETACHED"
Then Action26
    is confirmed.
    And 0.17 is assigned to FRACpd
    And 0.25 is assigned to FRACdd
    And 0.33 is assigned to FRACcut
    And 0.25 is assigned to FRACit
    And (edsi*FRACpd)/EDSIMMnom is assigned to MMnompd
    And (edsi*FRACdd)/EDSIMMnom is assigned to MMnomdd
    And (edsi*FRACcut)/EDSIMMnom is assigned to MMnomcut
    And (edsi*FRACit)/EDSIMMnom is assigned to MMnomit

RULE : Rule 20
If
    edsi is greater than 32000.0
    And edsi is less than or equal to 128000.0
    And |MODE|.Name is "SEMIDETACHED"
Then Action27
    is confirmed.
    And 0.17 is assigned to FRACpd
    And 0.24 is assigned to FRACdd
    And 0.31 is assigned to FRACcut
    And 0.28 is assigned to FRACit
    And (edsi*FRACpd)/EDSIMMnom is assigned to MMnompd
    And (edsi*FRACdd)/EDSIMMnom is assigned to MMnomdd
    And (edsi*FRACcut)/EDSIMMnom is assigned to MMnomcut
    And (edsi*FRACit)/EDSIMMnom is assigned to MMnomit

RULE : Rule 21
If
    edsi is greater than 128000.0
    And |MODE|.Name is "SEMIDETACHED"
Then Action28
    is confirmed.
    And 0.17 is assigned to FRACpd
    And 0.23 is assigned to FRACdd
    And 0.29 is assigned to FRACcut

```
RULE : Rule 21 (cont.)
    And 0.31 is assigned to FRACit
    And (edsi*FRACpd)/EDSIMMnom is assigned to MMnompd
    And (edsi*FRACdd)/EDSIMMnom is assigned to MMnomdd
    And (edsi*FRACcut)/EDSIMMnom is assigned to MMnomcut
    And (edsi*FRACit)/EDSIMMnom is assigned to MMnomit

RULE : Rule 22
If
    edsi is less than or equal to 2000.0
    And |MODE|.Name is "EMBEDDED"
Then Action29
    is confirmed.
    And 0.18 is assigned to FRACpd
RULE : Rule 22 (cont.)
    And 0.28 is assigned to FRACdd
    And 0.32 is assigned to FRACcut
    And 0.22 is assigned to FRACit
    And (edsi*FRACpd)/EDSIMMnom is assigned to MMnompd
    And (edsi*FRACdd)/EDSIMMnom is assigned to MMnomdd
    And (edsi*FRACcut)/EDSIMMnom is assigned to MMnomcut
    And (edsi*FRACit)/EDSIMMnom is assigned to MMnomit

RULE : Rule 23
If
    there is evidence of |SUBSYSTEM|.Requirement
    And there is evidence of |MODULE|.Requirement
    And there is evidence of |MODULE|.Modify
Then Action3
    is confirmed.
    And |MODULE|.EDSI is assigned to edsi
    And |MODULE|.EDSID is assigned to edsid
    And |MODULE|.EDSIC is assigned to edsic
    And |MODULE|.EDSII is assigned to edsii
    And (0.4*edsid)+(0.3*edsic)+(0.3*edsii) is assigned to |MODULE|.AAF
    And |MODULE|.AAF is assigned to aaf
    And edsi*aaf/100.0 is assigned to edsi
    And edsi+subedsi is assigned to subedsi
    And p+1.0 is assigned to p
    And Create Object 'MODULE_'\p\ |MODULE|
    And edsid is assigned to <|MODULE|>.EDSID
    And edsic is assigned to <|MODULE|>.EDSIC
    And edsii is assigned to <|MODULE|>.EDSII
    And aaf is assigned to <|MODULE|>.AAF
    And edsi is assigned to <|MODULE|>.EDSI
    And p is assigned to <|MODULE|>.Number
    And Reset |MODULE|.Requirement
```

RULE : Rule 24
If
    edsi is greater than 2000.0
    And edsi is less than or equal to 8000.0
    And |MODE|.Name is "EMBEDDED"
Then Action30
    is confirmed.
    And 0.18 is assigned to FRACpd
    And 0.27 is assigned to FRACdd
    And 0.3 is assigned to FRACcut
    And 0.25 is assigned to FRACit
    And (edsi*FRACpd)/EDSIMMnom is assigned to MMnompd
    And (edsi*FRACdd)/EDSIMMnom is assigned to MMnomdd
    And (edsi*FRACcut)/EDSIMMnom is assigned to MMnomcut
    And (edsi*FRACit)/EDSIMMnom is assigned to MMnomit

RULE : Rule 25
If
    edsi is greater than 8000.0
    And edsi is less than or equal to 32000.0
    And |MODE|.Name is "EMBEDDED"
Then Action31
    is confirmed.
    And 0.18 is assigned to FRACpd
    And 0.26 is assigned to FRACdd
    And 0.28 is assigned to FRACcut
    And 0.28 is assigned to FRACit
    And (edsi*FRACpd)/EDSIMMnom is assigned to MMnompd
    And (edsi*FRACdd)/EDSIMMnom is assigned to MMnomdd
    And (edsi*FRACcut)/EDSIMMnom is assigned to MMnomcut
    And (edsi*FRACit)/EDSIMMnom is assigned to MMnomit

RULE : Rule 26
If
    edsi is greater than 32000.0
    And edsi is less than or equal to 128000.0
    And |MODE|.Name is "EMBEDDED"
Then Action32
    is confirmed.
    And 0.18 is assigned to FRACpd
    And 0.25 is assigned to FRACdd
    And 0.26 is assigned to FRACcut
    And 0.31 is assigned to FRACit
    And (edsi*FRACpd)/EDSIMMnom is assigned to MMnompd
    And (edsi*FRACdd)/EDSIMMnom is assigned to MMnomdd
    And (edsi*FRACcut)/EDSIMMnom is assigned to MMnomcut
    And (edsi*FRACit)/EDSIMMnom is assigned to MMnomit

```
RULE : Rule 27
If
    edsi is greater than 128000.0
    And |MODE|.Name is "EMBEDDED"
Then Action33
    is confirmed.
    And 0.18 is assigned to FRACpd
    And 0.24 is assigned to FRACdd
    And 0.24 is assigned to FRACcut
    And 0.34 is assigned to FRACit
    And (edsi*FRACpd)/EDSIMMnom is assigned to MMnompd
    And (edsi*FRACdd)/EDSIMMnom is assigned to MMnomdd
    And (edsi*FRACcut)/EDSIMMnom is assigned to MMnomcut
    And (edsi*FRACit)/EDSIMMnom is assigned to MMnomit

RULE : Rule 28
If
    MMnomit is greater than 0.0
Then ACTION34
    is confirmed.
    And |MODULE|.PCAP is assigned to pcap
    And |MODULE|.LEXP is assigned to lexp
    And |MODULE|.VEXP is assigned to vexp

RULE : Rule 29
If
    pcap is less than or equal to 15.0
Then Action35
    is confirmed.
    And |MODULE|.PCAP_Rating is set to "Very_Low"
    And 1.0 is assigned to <|MODULE|>.PCAPpd
    And 1.5 is assigned to <|MODULE|>.PCAPdd
    And 1.5 is assigned to <|MODULE|>.PCAPcut
    And 1.5 is assigned to <|MODULE|>.PCAPit

RULE : Rule 30
If
    pcap is greater than 15.0
    And pcap is less than or equal to 35.0
Then Action36
    is confirmed.
    And |MODULE|.PCAP_Rating is set to "Low"
    And 1.0 is assigned to <|MODULE|>.PCAPpd
    And 1.2 is assigned to <|MODULE|>.PCAPdd
    And 1.2 is assigned to <|MODULE|>.PCAPcut
    And 1.2 is assigned to <|MODULE|>.PCAPit

RULE : Rule 31
If
    pcap is greater than 35.0
    And pcap is less than or equal to 55.0
```

```
RULE : Rule 31 (cont.)
Then Action37
    is confirmed.
    And |MODULE|.PCAP_Rating is set to "Nominal"
    And 1.0 is assigned to <|MODULE|>.PCAPpd
    And 1.0 is assigned to <|MODULE|>.PCAPdd
    And 1.0 is assigned to <|MODULE|>.PCAPcut
    And 1.0 is assigned to <|MODULE|>.PCAPit

RULE : Rule 32
If
    pcap is less than or equal to 75.0
    And pcap is greater than 55.0
Then Action38
    is confirmed.
    And |MODULE|.PCAP_Rating is set to "High"
    And 1.0 is assigned to <|MODULE|>.PCAPpd
    And 0.83 is assigned to <|MODULE|>.PCAPdd
    And 0.83 is assigned to <|MODULE|>.PCAPcut
    And 0.83 is assigned to <|MODULE|>.PCAPit

RULE : Rule 33
If
    pcap is less than or equal to 100.0
    And pcap is greater than 75.0
Then Action39
    is confirmed.
    And |MODULE|.PCAP_Rating is set to "Very_High"
    And 1.0 is assigned to <|MODULE|>.PCAPpd
    And 0.65 is assigned to <|MODULE|>.PCAPdd
    And 0.65 is assigned to <|MODULE|>.PCAPcut
    And 0.65 is assigned to <|MODULE|>.PCAPit

RULE : Rule 34
If
    lexp is less than or equal to 1.0
Then Action40
    is confirmed.
    And |MODULE|.LEXP_Rating is set to "Very_Low"
    And 1.02 is assigned to <|MODULE|>.LEXPpd
    And 1.1 is assigned to <|MODULE|>.LEXPdd
    And 1.2 is assigned to <|MODULE|>.LEXPcut
    And 1.2 is assigned to <|MODULE|>.LEXPit

RULE : Rule 35
If
    lexp is less than or equal to 4.0
    And lexp is greater than 1.0
Then Action41
    is confirmed.
    And |MODULE|.LEXP_Rating is set to "Low"
```

```
RULE : Rule 35 (cont.)
    And 1.0 is assigned to <|MODULE|>.LEXPpd
    And 1.05 is assigned to <|MODULE|>.LEXPdd
    And 1.1 is assigned to <|MODULE|>.LEXPcut
    And 1.1 is assigned to <|MODULE|>.LEXPit

RULE : Rule 36
If
    lexp is less than or equal to 12.0
    And lexp is greater than 4.0
Then Action42
    is confirmed.
    And |MODULE|.LEXP_Rating is set to "Nominal"
    And 1.0 is assigned to <|MODULE|>.LEXPpd
    And 1.0 is assigned to <|MODULE|>.LEXPdd
    And 1.0 is assigned to <|MODULE|>.LEXPcut
    And 1.0 is assigned to <|MODULE|>.LEXPit

RULE : Rule 37
If
    lexp is greater than 12.0
Then Action43
    is confirmed.
    And |MODULE|.LEXP_Rating is set to "High"
    And 1.0 is assigned to <|MODULE|>.LEXPpd
    And 0.98 is assigned to <|MODULE|>.LEXPdd
    And 0.92 is assigned to <|MODULE|>.LEXPcut
    And 0.92 is assigned to <|MODULE|>.LEXPit

RULE : Rule 38
If
    vexp is less than or equal to 4.0
Then Action44
    is confirmed.
    And |MODULE|.VEXP_Rating is set to "Very_Low"
    And 1.1 is assigned to <|MODULE|>.VEXPpd
    And 1.1 is assigned to <|MODULE|>.VEXPdd
    And 1.3 is assigned to <|MODULE|>.VEXPcut
    And 1.3 is assigned to <|MODULE|>.VEXPit

RULE : Rule 39
If
    vexp is greater than 4.0
    And vexp is less than or equal to 12.0
Then Action45
    is confirmed.
    And |MODULE|.VEXP_Rating is set to "Low"
    And 1.05 is assigned to <|MODULE|>.VEXPpd
    And 1.05 is assigned to <|MODULE|>.VEXPdd
    And 1.15 is assigned to <|MODULE|>.VEXPcut
    And 1.15 is assigned to <|MODULE|>.VEXPit
```

RULE : Rule 40
If
vexp is greater than 12.0
And vexp is less than or equal to 36.0
Then Action46
is confirmed.
And |MODULE|.VEXP_Rating is set to "Nominal"
And 1.0 is assigned to <|MODULE|>.VEXPpd
And 1.0 is assigned to <|MODULE|>.VEXPdd
And 1.0 is assigned to <|MODULE|>.VEXPcut
And 1.0 is assigned to <|MODULE|>.VEXPit

RULE : Rule 41
If
vexp is greater than 36.0
Then Action47
is confirmed.
And |MODULE|.VEXP_Rating is set to "High"
And 0.9 is assigned to <|MODULE|>.VEXPpd
And 0.9 is assigned to <|MODULE|>.VEXPdd
And 0.9 is assigned to <|MODULE|>.VEXPcut
And 0.9 is assigned to <|MODULE|>.VEXPit

RULE : Rule 42
If
edsi is greater than 0.0
Then Action48
is confirmed.
And |MODULE|.CPLXnesting is assigned to nesting
And |MODULE|.CPLXstdmathrou is assigned to stdmathrou
And |MODULE|.CPLXio is assigned to io
And |MODULE|.CPLXdatamgt is assigned to datamgt
And nesting+stdmathrou+io+datamgt is assigned to cplx

RULE : Rule 43
If
cplx is less than or equal to 5.0
Then Action49
is confirmed.
And |MODULE|.CPLX_Rating is set to "Very_Low"
And 0.7 is assigned to <|MODULE|>.CPLXpd
And 0.7 is assigned to <|MODULE|>.CPLXdd
And 0.7 is assigned to <|MODULE|>.CPLXcut
And 0.7 is assigned to <|MODULE|>.CPLXit

RULE : Rule 44
If
there is evidence of |SUBSYSTEM|.Requirement
And there is evidence of |MODULE|.Requirement
And there is no evidence of |MODULE|.Modify

```
RULE : Rule 44 (cont.)
Then Action5
    is confirmed.
    And |MODULE|.EDSI is assigned to edsi
    And edsi+subedsi is assigned to subedsi
    And p+1.0 is assigned to p
    And Create Object 'MODULE_'\p\ |MODULE|
    And p is assigned to <|MODULE|>.Number
    And edsi is assigned to <|MODULE|>.EDSI
    And Reset |MODULE|.Requirement

RULE : Rule 45
If
    cplx is less than or equal to 10.0
    And cplx is greater than 5.0
Then Action50
    is confirmed.
    And |MODULE|.CPLX_Rating is set to "Low"
    And 0.85 is assigned to <|MODULE|>.CPLXpd
    And 0.85 is assigned to <|MODULE|>.CPLXdd
    And 0.85 is assigned to <|MODULE|>.CPLXcut
    And 0.85 is assigned to <|MODULE|>.CPLXit

RULE : Rule 46
If
    cplx is less than or equal to 15.0
    And cplx is greater than 10.0
Then Action51
    is confirmed.
    And |MODULE|.CPLX_Rating is set to "Nominal"
    And 1.0 is assigned to <|MODULE|>.CPLXpd
    And 1.0 is assigned to <|MODULE|>.CPLXdd
    And 1.0 is assigned to <|MODULE|>.CPLXcut
    And 1.0 is assigned to <|MODULE|>.CPLXit

RULE : Rule 47
If
    cplx is less than or equal to 20.0
    And cplx is greater than 15.0
Then Action52
    is confirmed.
    And |MODULE|.CPLX_Rating is set to "High"
    And 1.15 is assigned to <|MODULE|>.CPLXpd
    And 1.15 is assigned to <|MODULE|>.CPLXdd
    And 1.15 is assigned to <|MODULE|>.CPLXcut
    And 1.15 is assigned to <|MODULE|>.CPLXit

RULE : Rule 48
If
    cplx is less than or equal to 30.0
    And cplx is greater than 20.0
```

RULE : Rule 48 (cont.)
Then Action53
is confirmed.
And |MODULE|.CPLX_Rating is set to "Very_High"
And 1.3 is assigned to <|MODULE|>.CPLXpd
And 1.3 is assigned to <|MODULE|>.CPLXdd
And 1.3 is assigned to <|MODULE|>.CPLXcut
And 1.3 is assigned to <|MODULE|>.CPLXit

RULE : Rule 49
If
cplx is less than or equal to 40.0
And cplx is greater than 30.0
Then Action54
is confirmed.
And |MODULE|.CPLX_Rating is set to "Extra_High"
And 1.65 is assigned to <|MODULE|>.CPLXpd
And 1.65 is assigned to <|MODULE|>.CPLXdd
And 1.65 is assigned to <|MODULE|>.CPLXcut
And 1.65 is assigned to <|MODULE|>.CPLXit

RULE : Rule 50
If
edsi is greater than 0.0
Then Action55
is confirmed.
And<|MODULE|>.CPLXpd*<|MODULE|>.LEXPpd*<|MODULE|>.VEXPpd*<|MODULE|>.PCAPpd is assigned to EAFmpd
And <|MODULE|>.CPLXdd*<|MODULE|>.LEXPdd*<|MODULE|>.VEXPdd*<|MODULE|>.PCAPddis assigned to EAFmdd
And <|MODULE|>.CPLXcut*<|MODULE|>.LEXPcut*<|MODULE|>.VEXPcut*<|MODULE|>.PCAPcutis assigned to EAFmcut
And <|MODULE|>.CPLXit*<|MODULE|>.LEXPit*<|MODULE|>.VEXPit*<|MODULE|>.PCAPitis assigned to EAFmit
And MMnompd*EAFmpd is assigned to <|SUBSYSTEM|>.MMmodpd
And MMnomdd*EAFmdd is assigned to <|SUBSYSTEM|>.MMmoddd
And MMnomcut*EAFmcut is assigned to <|SUBSYSTEM|>.MMmodcut
And MMnomit*EAFmit is assigned to <|SUBSYSTEM|>.MMmodit

RULE : Rule 51
If
|SUBSYSTEM|.Reliability is "Minimal_recovery_cost"
Then Action56
is confirmed.
And 0.8 is assigned to <|SUBSYSTEM|>.RELYpd
And 0.8 is assigned to <|SUBSYSTEM|>.RELYdd
And 0.8 is assigned to <|SUBSYSTEM|>.RELYcut
And 0.6 is assigned to <|SUBSYSTEM|>.RELYit
And <|SUBSYSTEM|>.Rely_rating is set to "Very_Low"

RULE : Rule 52
If
|SUBSYSTEM|.Reliability is "Low_recovery_cost"
Then Action57
is confirmed.
And 0.9 is assigned to <|SUBSYSTEM|>.RELYpd
And 0.9 is assigned to <|SUBSYSTEM|>.RELYdd
And 0.9 is assigned to <|SUBSYSTEM|>.RELYcut
And 0.8 is assigned to <|SUBSYSTEM|>.RELYit
And <|SUBSYSTEM|>.Rely_rating is set to "Low"

RULE : Rule 53
If
|SUBSYSTEM|.Reliability is "Moderate_recovery_cost"
Then Action58
is confirmed.
And 1.0 is assigned to <|SUBSYSTEM|>.RELYpd
And 1.0 is assigned to <|SUBSYSTEM|>.RELYdd
And 1.0 is assigned to <|SUBSYSTEM|>.RELYcut
And 1.0 is assigned to <|SUBSYSTEM|>.RELYit
And <|SUBSYSTEM|>.Rely_rating is set to "Nominal"

RULE : Rule 54
If
|SUBSYSTEM|.Reliability is "High_recovery_cost"
Then Action59
is confirmed.
And 1.1 is assigned to <|SUBSYSTEM|>.RELYpd
And 1.1 is assigned to <|SUBSYSTEM|>.RELYdd
And 1.1 is assigned to <|SUBSYSTEM|>.RELYcut
And 1.3 is assigned to <|SUBSYSTEM|>.RELYit
And <|SUBSYSTEM|>.Rely_rating is set to "High"

RULE : Rule 55
If
|SUBSYSTEM|.Reliability is "Life_threatening"
Then Action60
is confirmed.
And 1.3 is assigned to <|SUBSYSTEM|>.RELYpd
And 1.3 is assigned to <|SUBSYSTEM|>.RELYdd
And 1.3 is assigned to <|SUBSYSTEM|>.RELYcut
And 1.7 is assigned to <|SUBSYSTEM|>.RELYit
And <|SUBSYSTEM|>.Rely_rating is set to "Very_High"

RULE : Rule 56
If
|SUBSYSTEM|.Time is less than or equal to 50.0
Then Action61
is confirmed.
And 1.0 is assigned to <|SUBSYSTEM|>.TIMEpd
And 1.0 is assigned to <|SUBSYSTEM|>.TIMEdd

```
RULE : Rule 56 (cont.)
    And 1.0 is assigned to <|SUBSYSTEM|>.TIMEcut
    And 1.0 is assigned to <|SUBSYSTEM|>.TIMEit
    And <|SUBSYSTEM|>.Time_rating is set to "Nominal"

RULE : Rule 57
If
    |SUBSYSTEM|.Time is less than or equal to 70.0
    And |SUBSYSTEM|.Time is greater than 50.0
Then Action62
    is confirmed.
    And 1.1 is assigned to <|SUBSYSTEM|>.TIMEpd
    And 1.1 is assigned to <|SUBSYSTEM|>.TIMEdd
    And 1.1 is assigned to <|SUBSYSTEM|>.TIMEcut
    And 1.15 is assigned to <|SUBSYSTEM|>.TIMEit
    And <|SUBSYSTEM|>.Time_rating is set to "High"

RULE : Rule 58
If
    |SUBSYSTEM|.Time is less than or equal to 94.0
    And |SUBSYSTEM|.Time is greater than 70.0
Then Action63
    is confirmed.
    And 1.3 is assigned to <|SUBSYSTEM|>.TIMEpd
    And 1.25 is assigned to <|SUBSYSTEM|>.TIMEdd
    And 1.25 is assigned to <|SUBSYSTEM|>.TIMEcut
    And 1.4 is assigned to <|SUBSYSTEM|>.TIMEit
    And <|SUBSYSTEM|>.Time_rating is set to "Very_High"

RULE : Rule 59
If
    |SUBSYSTEM|.Time is greater than or equal to 95.0
Then Action64
    is confirmed.
    And 1.65 is assigned to <|SUBSYSTEM|>.TIMEpd
    And 1.55 is assigned to <|SUBSYSTEM|>.TIMEdd
    And 1.55 is assigned to <|SUBSYSTEM|>.TIMEcut
    And 1.95 is assigned to <|SUBSYSTEM|>.TIMEit
    And <|SUBSYSTEM|>.Time_rating is set to "Extra_High"

RULE : Rule 60
If
    |SUBSYSTEM|.Stor is less than or equal to 50.0
Then Action65
    is confirmed.
    And 1.0 is assigned to <|SUBSYSTEM|>.STORpd
    And 1.0 is assigned to <|SUBSYSTEM|>.STORdd
    And 1.0 is assigned to <|SUBSYSTEM|>.STORcut
    And 1.0 is assigned to <|SUBSYSTEM|>.STORit
    And <|SUBSYSTEM|>.Stor_rating is set to "Nominal"
```

```
RULE : Rule 61
If
    |SUBSYSTEM|.Stor is less than or equal to 70.0
    And |SUBSYSTEM|.Stor is greater than 50.0
Then Action66
    is confirmed.
    And 1.05 is assigned to <|SUBSYSTEM|>.STORpd
    And 1.05 is assigned to <|SUBSYSTEM|>.STORdd
    And 1.05 is assigned to <|SUBSYSTEM|>.STORcut
    And 1.1 is assigned to <|SUBSYSTEM|>.STORit
    And <|SUBSYSTEM|>.Stor_rating is set to "High"

RULE : Rule 62
If
    |SUBSYSTEM|.Stor is less than or equal to 94.0
    And |SUBSYSTEM|.Stor is greater than 70.0
Then Action67
    is confirmed.
    And 1.2 is assigned to <|SUBSYSTEM|>.STORpd
    And 1.15 is assigned to <|SUBSYSTEM|>.STORdd
    And 1.15 is assigned to <|SUBSYSTEM|>.STORcut
    And 1.35 is assigned to <|SUBSYSTEM|>.STORit
    And <|SUBSYSTEM|>.Stor_rating is set to "Very_High"

RULE : Rule 63
If
    |SUBSYSTEM|.Stor is greater than 94.0
Then Action68
    is confirmed.
    And 1.55 is assigned to <|SUBSYSTEM|>.STORpd
    And 1.45 is assigned to <|SUBSYSTEM|>.STORdd
    And 1.45 is assigned to <|SUBSYSTEM|>.STORcut
    And 1.85 is assigned to <|SUBSYSTEM|>.STORit
    And <|SUBSYSTEM|>.Stor_rating is set to "Extra_High"

RULE : Rule 64
If
    |SUBSYSTEM|.Virt is "Once_a_year"
Then Action69
    is confirmed.
    And 0.95 is assigned to <|SUBSYSTEM|>.VIRTpd
    And 0.9 is assigned to <|SUBSYSTEM|>.VIRTdd
    And 0.85 is assigned to <|SUBSYSTEM|>.VIRTcut
    And 0.8 is assigned to <|SUBSYSTEM|>.VIRTit
    And <|SUBSYSTEM|>.Stor_rating is set to "Low"

RULE : Rule 65
If
    |SUBSYSTEM|.Virt is "Every_6_months"
Then Action70
    is confirmed.
```

RULE : Rule 65 (cont.)
And 1.0 is assigned to <|SUBSYSTEM|>.VIRTpd
And 1.0 is assigned to <|SUBSYSTEM|>.VIRTdd
And 1.0 is assigned to <|SUBSYSTEM|>.VIRTcut
And 1.0 is assigned to <|SUBSYSTEM|>.VIRTit
And <|SUBSYSTEM|>.Stor_rating is set to "Nominal"

RULE : Rule 66
If
|SUBSYSTEM|.Virt is "Every_2_months"
Then Action71
is confirmed.
And 1.1 is assigned to <|SUBSYSTEM|>.VIRTpd
And 1.12 is assigned to <|SUBSYSTEM|>.VIRTdd
And 1.15 is assigned to <|SUBSYSTEM|>.VIRTcut
And 1.2 is assigned to <|SUBSYSTEM|>.VIRTit
And <|SUBSYSTEM|>.Stor_rating is set to "High"

RULE : Rule 67
If
|SUBSYSTEM|.Virt is "Every_2_weeks"
Then Action72
is confirmed.
And 1.2 is assigned to <|SUBSYSTEM|>.VIRTpd
And 1.25 is assigned to <|SUBSYSTEM|>.VIRTdd
And 1.3 is assigned to <|SUBSYSTEM|>.VIRTcut
And 1.4 is assigned to <|SUBSYSTEM|>.VIRTit
And <|SUBSYSTEM|>.Stor_rating is set to "Very_High"

RULE : Rule 68
If
|SUBSYSTEM|.Turn is "Interactive"
Then Action73
is confirmed.
And 0.98 is assigned to <|SUBSYSTEM|>.TURNpd
And 0.95 is assigned to <|SUBSYSTEM|>.TURNdd
And 0.7 is assigned to <|SUBSYSTEM|>.TURNcut
And 0.9 is assigned to <|SUBSYSTEM|>.TURNit
And <|SUBSYSTEM|>.Turn_rating is set to "Low"

RULE : Rule 69
If
|SUBSYSTEM|.Turn is "Less_than_4_hours"
Then Action74
is confirmed.
And 1.0 is assigned to <|SUBSYSTEM|>.TURNpd
And 1.0 is assigned to <|SUBSYSTEM|>.TURNdd
And 1.0 is assigned to <|SUBSYSTEM|>.TURNcut
And 1.0 is assigned to <|SUBSYSTEM|>.TURNit
And <|SUBSYSTEM|>.Turn_rating is set to "Nominal"

```
RULE : Rule 70
If
    |SUBSYSTEM|.Turn is "Between_4_and_12_hours"
Then Action75
    is confirmed.
    And 1.0 is assigned to <|SUBSYSTEM|>.TURNpd
    And 1.0 is assigned to <|SUBSYSTEM|>.TURNdd
    And 1.1 is assigned to <|SUBSYSTEM|>.TURNcut
    And 1.15 is assigned to <|SUBSYSTEM|>.TURNit
    And <|SUBSYSTEM|>.Turn_rating is set to "High"

RULE : Rule 71
If
    |SUBSYSTEM|.Turn is "Greater_than_12_hours"
Then Action76
    is confirmed.
    And 1.02 is assigned to <|SUBSYSTEM|>.TURNpd
    And 1.05 is assigned to <|SUBSYSTEM|>.TURNdd
    And 1.2 is assigned to <|SUBSYSTEM|>.TURNcut
    And 1.3 is assigned to <|SUBSYSTEM|>.TURNit
    And <|SUBSYSTEM|>.Turn_rating is set to "Very_High"

RULE : Rule 72
If
    |SUBSYSTEM|.ACAP is "Lowest_15th_percentile"
Then Action77
    is confirmed.
    And 1.8 is assigned to <|SUBSYSTEM|>.ACAPpd
    And 1.35 is assigned to <|SUBSYSTEM|>.ACAPdd
    And 1.35 is assigned to <|SUBSYSTEM|>.ACAPcut
    And 1.5 is assigned to <|SUBSYSTEM|>.ACAPit
    And <|SUBSYSTEM|>.ACAP_rating is set to "Very_Low"

RULE : Rule 73
If
    |SUBSYSTEM|.ACAP is "Between_the_15th_and_35th_percentile"
Then Action78
    is confirmed.
    And 1.35 is assigned to <|SUBSYSTEM|>.ACAPpd
    And 1.15 is assigned to <|SUBSYSTEM|>.ACAPdd
    And 1.15 is assigned to <|SUBSYSTEM|>.ACAPcut
    And 1.2 is assigned to <|SUBSYSTEM|>.ACAPit
    And <|SUBSYSTEM|>.ACAP_rating is set to "Low"

RULE : Rule 74
If
    |SUBSYSTEM|.ACAP is "Between_the_35th_and_55th_percentile"
Then Action79
    is confirmed.
    And 1.0 is assigned to <|SUBSYSTEM|>.ACAPpd
    And 1.0 is assigned to <|SUBSYSTEM|>.ACAPdd
```

RULE : Rule 74 (cont.)
And 1.0 is assigned to <|SUBSYSTEM|>.ACAPcut
And 1.0 is assigned to <|SUBSYSTEM|>.ACAPit
And <|SUBSYSTEM|>.ACAP_rating is set to "Nominal"

RULE : Rule 75
If
|SUBSYSTEM|.ACAP is "Between_the_55th_and_75th_percentile"
Then Action80
is confirmed.
And 0.75 is assigned to <|SUBSYSTEM|>.ACAPpd
And 0.9 is assigned to <|SUBSYSTEM|>.ACAPdd
And 0.9 is assigned to <|SUBSYSTEM|>.ACAPcut
And 0.85 is assigned to <|SUBSYSTEM|>.ACAPit
And <|SUBSYSTEM|>.ACAP_rating is set to "High"

RULE : Rule 76
If
|SUBSYSTEM|.ACAP is "Above_the_75th_percentile"
Then Action81
is confirmed.
And 0.55 is assigned to <|SUBSYSTEM|>.ACAPpd
And 0.75 is assigned to <|SUBSYSTEM|>.ACAPdd
And 0.75 is assigned to <|SUBSYSTEM|>.ACAPcut
And 0.7 is assigned to <|SUBSYSTEM|>.ACAPit
And <|SUBSYSTEM|>.ACAP_rating is set to "Very_High"

RULE : Rule 77
If
|SUBSYSTEM|.AEXP is "Less_than_4_months"
Then Action82
is confirmed.
And 1.4 is assigned to <|SUBSYSTEM|>.AEXPpd
And 1.3 is assigned to <|SUBSYSTEM|>.AEXPdd
And 1.25 is assigned to <|SUBSYSTEM|>.AEXPcut
And 1.25 is assigned to <|SUBSYSTEM|>.AEXPit
And <|SUBSYSTEM|>.AEXP_rating is set to "Very_Low"

RULE : Rule 78
If
|SUBSYSTEM|.AEXP is "Between_4_and_12_months"
Then Action83
is confirmed.
And 1.2 is assigned to <|SUBSYSTEM|>.AEXPpd
And 1.15 is assigned to <|SUBSYSTEM|>.AEXPdd
And 1.1 is assigned to <|SUBSYSTEM|>.AEXPcut
And 1.1 is assigned to <|SUBSYSTEM|>.AEXPit
And <|SUBSYSTEM|>.AEXP_rating is set to "Low"

RULE : Rule 79
If
    |SUBSYSTEM|.AEXP is "Between_1_and_3_years"
Then Action84
    is confirmed.
    And 1.0 is assigned to <|SUBSYSTEM|>.AEXPpd
    And 1.0 is assigned to <|SUBSYSTEM|>.AEXPdd
    And 1.0 is assigned to <|SUBSYSTEM|>.AEXPcut
    And 1.0 is assigned to <|SUBSYSTEM|>.AEXPit
    And <|SUBSYSTEM|>.AEXP_rating is set to "Nominal"

RULE : Rule 80
If
    |SUBSYSTEM|.AEXP is "Between_3_and_6_years"
Then Action85
    is confirmed.
    And 0.87 is assigned to <|SUBSYSTEM|>.AEXPpd
    And 0.9 is assigned to <|SUBSYSTEM|>.AEXPdd
    And 0.92 is assigned to <|SUBSYSTEM|>.AEXPcut
    And 0.92 is assigned to <|SUBSYSTEM|>.AEXPit
    And <|SUBSYSTEM|>.AEXP_rating is set to "High"

RULE : Rule 81
If
    |SUBSYSTEM|.AEXP is "Greater_than_6_years"
Then Action86
    is confirmed.
    And 0.75 is assigned to <|SUBSYSTEM|>.AEXPpd
    And 0.8 is assigned to <|SUBSYSTEM|>.AEXPdd
    And 0.85 is assigned to <|SUBSYSTEM|>.AEXPcut
    And 0.85 is assigned to <|SUBSYSTEM|>.AEXPit
    And <|SUBSYSTEM|>.AEXP_rating is set to "Very_High"

RULE : Rule 82
If
    |SUBSYSTEM|.MODP is "No_use_of_MPPs"
Then Action87
    is confirmed.
    And 1.05 is assigned to <|SUBSYSTEM|>.MODPpd
    And 1.1 is assigned to <|SUBSYSTEM|>.MODPdd
    And 1.25 is assigned to <|SUBSYSTEM|>.MODPcut
    And 1.5 is assigned to <|SUBSYSTEM|>.MODPit
    And <|SUBSYSTEM|>.MODP_rating is set to "Very_low"

RULE : Rule 83
If
    |SUBSYSTEM|.MODP is "Beginning_use_of_MPPs"
Then Action88
    is confirmed.
    And 1.0 is assigned to <|SUBSYSTEM|>.MODPpd
    And 1.05 is assigned to <|SUBSYSTEM|>.MODPdd

RULE : Rule 83 (cont.)
And 1.1 is assigned to <|SUBSYSTEM|>.MODPcut
And 1.2 is assigned to <|SUBSYSTEM|>.MODPit
And <|SUBSYSTEM|>.MODP_rating is set to "Low"

RULE : Rule 84
If
|SUBSYSTEM|.MODP is "Reasonably_experienced_in_some_MPP_use"
Then Action89
is confirmed.
And 1.0 is assigned to <|SUBSYSTEM|>.MODPpd
And 1.0 is assigned to <|SUBSYSTEM|>.MODPdd
And 1.0 is assigned to <|SUBSYSTEM|>.MODPcut
And 1.0 is assigned to <|SUBSYSTEM|>.MODPit
And <|SUBSYSTEM|>.MODP_rating is set to "Nominal"

RULE : Rule 85
If
|SUBSYSTEM|.MODP is "Reasonably_experienced_in_most_MPP_use"
Then Action90
is confirmed.
And 1.0 is assigned to <|SUBSYSTEM|>.MODPpd
And 0.95 is assigned to <|SUBSYSTEM|>.MODPdd
And 0.9 is assigned to <|SUBSYSTEM|>.MODPcut
And 0.83 is assigned to <|SUBSYSTEM|>.MODPit
And <|SUBSYSTEM|>.MODP_rating is set to "High"

RULE : Rule 86
If
|SUBSYSTEM|.MODP is "Routine_use_of_all_MPPs"
Then Action91
is confirmed.
And 1.0 is assigned to <|SUBSYSTEM|>.MODPpd
And 0.9 is assigned to <|SUBSYSTEM|>.MODPdd
And 0.8 is assigned to <|SUBSYSTEM|>.MODPcut
And 0.65 is assigned to <|SUBSYSTEM|>.MODPit
And <|SUBSYSTEM|>.MODP_rating is set to "Very_High"

RULE : Rule 87
If
|SUBSYSTEM|.Tool is "Basic_microcomputer_tools"
Then Action92
is confirmed.
And 1.02 is assigned to <|SUBSYSTEM|>.TOOLpd
And 1.05 is assigned to <|SUBSYSTEM|>.TOOLdd
And 1.35 is assigned to <|SUBSYSTEM|>.TOOLcut
And 1.45 is assigned to <|SUBSYSTEM|>.TOOLit
And <|SUBSYSTEM|>.Tool_rating is set to "Very_low"

```
RULE : Rule 88
If
    |SUBSYSTEM|.Tool is "Basic_minicomputer_tools"
Then Action93
    is confirmed.
    And 1.0 is assigned to <|SUBSYSTEM|>.TOOLpd
    And 1.05 is assigned to <|SUBSYSTEM|>.TOOLdd
    And 1.1 is assigned to <|SUBSYSTEM|>.TOOLcut
    And 1.2 is assigned to <|SUBSYSTEM|>.TOOLit
    And <|SUBSYSTEM|>.Tool_rating is set to "Low"

RULE : Rule 89
If
    |SUBSYSTEM|.Tool is "Strong_mini_or_basic_maxi_computer_tools"
Then Action94
    is confirmed.
    And 1.0 is assigned to <|SUBSYSTEM|>.TOOLpd
    And 1.0 is assigned to <|SUBSYSTEM|>.TOOLdd
    And 1.0 is assigned to <|SUBSYSTEM|>.TOOLcut
    And 1.0 is assigned to <|SUBSYSTEM|>.TOOLit
    And <|SUBSYSTEM|>.MODP_rating is set to "Nominal"

RULE : Rule 90
If
    |SUBSYSTEM|.Tool is "Strong_maxi_computer_tools"
Then Action95
    is confirmed.
    And 0.98 is assigned to <|SUBSYSTEM|>.TOOLpd
    And 0.95 is assigned to <|SUBSYSTEM|>.TOOLdd
    And 0.9 is assigned to <|SUBSYSTEM|>.TOOLcut
    And 0.85 is assigned to <|SUBSYSTEM|>.TOOLit
    And <|SUBSYSTEM|>.Tool_rating is set to "High"

RULE : Rule 91
If
    |SUBSYSTEM|.Tool is "Advanced_maxi_computer_tools"
Then Action96
    is confirmed.
    And 0.95 is assigned to <|SUBSYSTEM|>.TOOLpd
    And 0.9 is assigned to <|SUBSYSTEM|>.TOOLdd
    And 0.8 is assigned to <|SUBSYSTEM|>.TOOLcut
    And 0.7 is assigned to <|SUBSYSTEM|>.TOOLit
    And <|SUBSYSTEM|>.Tool_rating is set to "Very_High"

RULE : Rule 92
If
    |SUBSYSTEM|.Sced is "Severely_accelerated_development_schedule"
Then Action97
    is confirmed.
    And 1.1 is assigned to <|SUBSYSTEM|>.SCEDpd
    And 1.25 is assigned to <|SUBSYSTEM|>.SCEDdd
```

RULE : Rule 92 (cont.)
And 1.25 is assigned to <|SUBSYSTEM|>.SCEDcut
And 1.25 is assigned to <|SUBSYSTEM|>.SCEDit
And <|SUBSYSTEM|>.Sced_rating is set to "Very_Low"

RULE : Rule 93
If
|SUBSYSTEM|.Sced is "Accelerated_development_schedule"
Then Action98
is confirmed.
And 1.0 is assigned to <|SUBSYSTEM|>.SCEDpd
And 1.15 is assigned to <|SUBSYSTEM|>.SCEDdd
And 1.15 is assigned to <|SUBSYSTEM|>.SCEDcut
And 1.1 is assigned to <|SUBSYSTEM|>.SCEDit
And <|SUBSYSTEM|>.Sced_rating is set to "Low"

RULE : Rule 94
If
|SUBSYSTEM|.Sced is "Programmed_development_schedule"
Then Action99
is confirmed.
And 1.0 is assigned to <|SUBSYSTEM|>.SCEDpd
And 1.0 is assigned to <|SUBSYSTEM|>.SCEDdd
And 1.0 is assigned to <|SUBSYSTEM|>.SCEDcut
And 1.0 is assigned to <|SUBSYSTEM|>.SCEDit
And <|SUBSYSTEM|>.Sced_rating is set to "Nominal"

RULE : Rule 95
If
there is evidence of |SUBSYSTEM|.Requirement
And there is evidence of |MODULE|.Requirement
Then MODULE.Requirement
is confirmed.
And Reset |MODULE|.Modify
And Reset |MODULE|.EDSI
And Reset |MODULE|.EDSID
And Reset |MODULE|.EDSIC
And Reset |MODULE|.EDSII
And Reset |MODULE|.AAF

RULE : Rule 96
If
there is evidence of |SUBSYSTEM|.Requirement
Then SUBSYSTEM.Requirement
is confirmed.
And n+1.0 is assigned to n
And Create Object 'SUBSYSTEM_'\n\ |SUBSYSTEM|

## Bibliography


Allen, Maj Mary K. The Development of an Artificial Intelligence System for Inventory Management. PhD thesis, Graduate School of Ohio State University. Oak Brook IL: Council of Logistics Management, 1989.

Boehm, Barry W. Software Engineering Economics. Englewood Cliffs NJ: Prentice-Hall, 1981.

Ferens, Daniel V. Defense System Software Project Management. Wright-Patterson AFB OH: Air Force Institute of Technology, School of Systems and Logistics, January 1990.

-----. "Softcost 676, A Case Study for Software Cost Estimation." Class handout distributed in IMGT 676, Software Cost Estimation. School of Systems and Logistics, Air Force Institute of Technology, Wright-Patterson AFB OH, April 1990.

-----. Class handout distributed in IMGT 676, Software Cost Estimation. School of Systems and Logistics, Air Force Institute of Technology, Wright-Patterson AFB OH, April 1990.

Goodson, Capt James L. and lst Lt Greg Pshsnychniak. "Term Project for Software Cost Estimation." Report for IMGT 676, Software Cost Estimation. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, May 1990.

Hazen, Capt Christopher M. How Can Air Force Civil Engineers Use Expert Systems? MS thesis, AFIT/GEM/LSM/88S-9. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 1989 (AD-A201583).

Hicks, Richard and Ronald Lee. VP-Expert™ for Business Applications. Oakland CA: Holden-Day, Inc., 1988.

Holt, Lt Col James R. "Knowledge Engineering Outline." Class lecture in LOGM 592, Artificial Intelligence Applications for Managers, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 1990.

-----. Class lecture in LOGM 592, Artificial Intelligence Applications for Managers, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 1990.

McMurry, Capt Deanna L. and Capt Kenneth L. Nelson. "Software Cost Estimating Case." Report for IMGT 676, Software Cost Estimation. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, May 1990.

Nexpert Object™ Fundamentals. Neuron Data Inc., Palo Alto CA, 1989.

Nexpert™ Partners, Service Catalog. Neuron Data Inc., Palo Alto CA, 1989.

Prerau, David S. "Selection of an Appropriate Domain for an Expert System," The AI Magazine, Summer, 1985, pp. 27-30.

Price S. Reference Manual. General Electric Company, Cherry Hill NJ: 1989.

Project Officer's Guide to Life Cycle Cost (LCC) Models. Office for Computer Resources and Analysis, Deputy for Acquisition Logistics, Los Angels AFB CA, August 1988.

Rasmsus, Daniel. "The Expert Is In," MacUser, September, 1989, pp.136-160.

SEER™ System Evaluation and Estimation of Resources. SEER™-SEM User's Manual. Galorath Associates, Inc., Marina Del Rey CA, 1988.

Stepanek, Stephen J. Specification of Expert Systems (U), Volume II, Expert System Cost Accounting: Final Report, 15 November 1988. Contract F04701-88-C-0031. El Segundo CA: Tecolote Research, Inc, November 1988.

Valusek, Lt Col John R. Class lecture in OPR 620, Decision Support Systems. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, May 1990.

Waterman, Donald A. A Guide to Expert Systems. Reading MA: Addison-Wesley Publishing Company, Inc., 1976.

Young, Lawrence F. Decision Support and Idea Processing Systems. Dubuque IA: Wm. C. Brown Publishers, 1989.

VITA

Major James L. Goodson [redacted] graduated from high school and enlisted in the Air Force in 1967. After attending technical school at Lowry AFB, Colorado, he was stationed at U-tapao Royal Thai Air Base as a weapons technician. Thereafter, he was assigned to two more tours in South East Asia, flying as a gunner on AC-119 and AC-130 Gunships. In 1979, he graduated from Troy State University with a Bachelor of Science degree in Psychology.

Major Goodson was commissioned through the Officer Training School at Lackland AFB, Texas. He was then assigned to the Air Force Flight Test Center at Edwards AFB, California, as Chief of Maintenance for the F-16 Test Force, with a follow on job as Maintenance Supervisor for the Field Maintenance Squadron. In 1983, he received a Masters of Science Degree in Systems Management from the University of Southern California. Shortly thereafter, he was assigned to Chanute AFB, Illinois as an instructor in the Aircraft Maintenance Officer Course. While still at Chanute AFB, Maj Goodson was assigned as Chief of the Aircraft Systems Division. In 1986, he was assigned to Air Force Space Division at Los Angeles AFB, California, where he was assigned as the Chief of the Logistics Support Division for the Global Positioning Satellite System (GPS). There, his responsibilities included logistics management of the GPS ground segment and concurrently, the logistics development for the Ground/Airborne Integrated Terminal of the Nuclear Detonation Detection System.

[redacted] [redacted]

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*



| 1. AGENCY USE ONLY *(Leave blank)* | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | September 1990 | Master's Thesis |

**4. TITLE AND SUBTITLE**

THE DEVELOPMENT OF AN EXPERT SYSTEM FOR SOFTWARE COST ESTIMATION

**5. FUNDING NUMBERS**

**6. AUTHOR(S)**

James L. Goodson, Major, USAF

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

Air Force Institute of Technology
WPAFB OH 45433-6583

**8. PERFORMING ORGANIZATION REPORT NUMBER**

AFIT/GLM/LSM/90S-21

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

**11. SUPPLEMENTARY NOTES**

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**

Approved for public release; distribution unlimited

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** *(Maximum 200 words)* Software development has become an integral part of new weapon system design as sophisticated computer operating and control systems have proliferated. Concurrently, there has been an ever increasing requirement to accurately forecast system development cost. Experts use complicated software cost estimating tools which are difficult for novice analysts to use. Expert system technology appears to be able to bridge this gap. Policymakers have been intrigued by the potential offered bythe development of expert systems in today's weapon systems for system support and development and operational utility. The primary objective of this thesis was to gather expert knowledge in software cost estimation and integrate it with a powerful analytical software cost estimation algorithm. A battery of questions where given to the experts to elicit responses relative to their knowledge in software cost estimation. Responses where integrated with algorithms with the detailed COCOMO cost estimation model. The expert system designed in this research provides software developers, program managers, and cost analysts an easy mechanism for determining software development costs. It provides an intelligent preproccessor, numeric algorithms and an intelligent post processor in one tool. The expert system can help the novice make accurate estimates and speed the process for experts.

**14. SUBJECT TERMS**

Artificial Intelligence, Expert System, Software Cost Estimation

**15. NUMBER OF PAGES**

122

**16. PRICE CODE**

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | UL |